

MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

SERVERLESS APPROACH TO SECURITY AUTOMATION

LOADING AN OPENSsh HOSTKEY

FROM A HARDWARE TOKEN ON FREEBSD

OPENBSD 6.0:

WHY AND HOW

INSTALLING WINDOWS 10

USING VNC ON FREEBSD 11 AND ABOVE

**HOW TO CONNECT PYCHARM
TO DEBUG A REMOTE DOCKER CONTAINER**

USING THE CONTAINERS REMOTE INTERPRETER IN BSD

INTERVIEW WITH

**EMILE HEITOR, CTO AND CO-OWNER OF NBS SYSTEM,
HEAD OF RESEARCH & EXPERTISE DEPARTMENT OCEANET TECHNOLOGY**

VOL 10 NO 10

ISSUE 09/2016 (86)

1898-9144

Simplify your Data Center

Meet iXRack — a customized and repeatable rack-scale infrastructure ideal for web-scale, virtualization, big data, private cloud, and virtually any enterprise business application.



- Converged & Repeatable
- Customizable for VMware, Big Data, Cloud & Hyperscale
- Up to 70% Lower TCO Than AWS
- Scalable Deployment

For more information on iXRack, visit **ixsystems.com/iXRack** today.

Dear Readers,

FreeBSD 11.0 is finally out! Were you impatient and did you download your upgrade right away?

I'm writing to you from my hotel room in Orlando, Florida. Beautiful weather is outside my window, Hurricane Matthew hasn't reached this city. I hope you all are safe where you are and that you enjoy this autumn, whether it's cold and rainy or warm and sunny. We also have Halloween coming very soon. What do you think about this American tradition?

Issue-wise, let's begin with "Serverless Approach to Security Automation" by Renan Dias. Grab some coffee and fix your corrupted server with us.

Moving on to the FreeBSD Corner. Here you will find an article by Mike Tanca, "Loading an OpenSSH Hostkey From a Hardware Token on FreeBSD". What if someone steals a copy of your private key? What if someone breaks into your host and makes off with your hostkey? Find the solution with this article.

Next we will hop right into "Install Windows 10 using VNC on FreeBSD 11 and Above" by Trent Thompson. You have been asking us about more articles about bhyve - here it is.

We have also been asked about more articles on OpenBSD. We hope "OpenBSD 6.0: Why and How" by Derek Sivers will be interesting for you.

If you are a fan of Docker and debugging, we have you covered with Miguel Tavares' article in "How to Connect Pycharm to Debug a Remote Docker Container Using the Containers Remote Interpreter in BSD".

At the end of this issue, you will find an interview with Emile Heitor, CTO and Co-owner of NBS System, and Head of the Research & Expertise Department at Oceanet Technology as well as Rob's Column.

We hope you enjoy this issue and have a nice and sunny October.

Marta & BSD Team



Editor in Chief:

Marta Ziemianowicz

marta.ziemianowicz@software.com.pl

Contributing:

Trent Thompson, Renan Dias, Mike Tenesca, Derek Sivers, Miguel Tavares, Emile Heitor and Rob Somerville.

Top Betatesters & Proofreaders:

Denise Ebery, Eric Geissinger, Luca Ferrari, Imad Soltani, Olaoluwa Omo-kanwaye, Radjis Mahangoe, Mani Kanth and Mark VonFange.

Special Thanks:

Annie Zhang

Denise Ebery

DTP:

Marta Ziemianowicz

Senior Consultant/Publisher:

Paweł Marciniak

pawel@software.com.pl

CEO:

Joanna Kretowicz

joanna.kretowicz@software.com.pl

Publisher:

Hakin9 Media SK 02-676 Warsaw, Poland Postepu 17D Poland worldwide publishing editors@bsdmag.org www.bsdmag.org

Hakin9 Media SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail: editors@bsdmag.org.

All trademarks presented in the magazine were used only for informative purposes. All rights to trademarks presented in the magazine are reserved by the companies which own them.

CONTENTS

News

BSD World Monthly News 4

by Marta Ziemianowicz

This column presents the latest news coverage of events, product releases and trending topics.

Security

Serverless Approach to Security Automation 17

by Renan Dias

Everyone is talking about DevOps (Development and Operations). Some people say DevOps is the job of a single man, some other people say DevOps is rather a culture and is about the collaboration between the development team and the operations team. The truth is that, regardless of the conflicting ideas, everyone agrees that one of DevOps' main goals is automation. Software release automation, right? Well, not quite. There are a lot of things that can be automated: software release, infrastructure provisioning, testing, benchmarking and security, to name a few. Now, did you notice the last thing on this list? Security.

FreeBSD Corner

Loading an OpenSSH Hostkey From a Hardware Token on FreeBSD 23

by Mike Tanca

I had a requirement for creating an sftp server that needs strong client and host authentication. The host needs to know it's an authorized client connection, and the client needs to know it's really the host it's connecting to. SSH and public key crypto is great for this, but what if someone steals a copy of your private key? What if someone breaks into your host and makes off with your hostkey? Until you detect the compromise and revoke and regenerate keys, you run the risk of a man in the middle attack, among other things.

Installing Windows 10 using VNC on FreeBSD 11 and Above 37

by Trent Thompson

This October of 2016 will be a special month for FreeBSD virtualization. Not only will the most recent release of FreeBSD be ready, but it will have been a year since UEFI booting in bhyve was announced via the FreeBSD-Virtualization Mailing List. At the time, bhyve did not have the ability to allow for any type of graphical console, outside of something run on the guest OS like RDP, VNC, or SPICE. Instead, bhyve used a serial console as a means to communicate with the guest operating system.

OpenBSD

OpenBSD 6.0: Why and How 58

by Derek Sivers

The only operating system I use on my computers is not Mac, not Windows, and not even Linux. It's OpenBSD, and I love it so much.

Since OpenBSD 6.0 was released today, I figured I should say a little something about why I love it, and how you can try it.

Docker

How to Connect Pycharm to Debug a Remote Docker Container Using the Containers Remote Interpreter in BSD 70

by Miguel Tavares

For a little background on my activity, I've been working with Python and Stackless Python on Django MVC's on several BSD servers and using PyCharm as Python IDE to develop on.

Interview

Emile Heitor, CTO and Co-owner of NBS System, and Head of the Research & Expertise Department at Oceanet Technology 78

by Marta Ziemianowicz, Marta Sienicka & Marta Strzelec

Rob's Column 91

by Rob Somerville

BSD Certification

The BSD Certification Group Inc. (BSDCG) is a non-profit organization committed to creating and maintaining a global certification standard for system administration on BSD based operating systems.

? WHAT CERTIFICATIONS ARE AVAILABLE?

BSDA: Entry-level certification suited for candidates with a general Unix background and at least six months of experience with BSD systems.

BDSP: Advanced certification for senior system administrators with at least three years of experience on BSD systems. Successful BDSP candidates are able to demonstrate strong to expert skills in BSD Unix system administration.

✓ WHERE CAN I GET CERTIFIED?

We're pleased to announce that after 7 months of negotiations and the work required to make the exam available in a computer based format, that the BSDA exam is now available at several hundred testing centers around the world. Paper based BSDA exams cost \$75 USD. Computer based BSDA exams cost \$150 USD. The price of the BDSP exams are yet to be determined.

Payments are made through our registration website:
<https://register.bsdcertification.org/register/payment>

i WHERE CAN I GET MORE INFORMATION?

More information and links to our mailing lists, LinkedIn groups, and Facebook group are available at our website:
<http://www.bsdcertification.org>

Registration for upcoming exam events is available at our registration website:
<https://register.bsdcertification.org/register/get-a-bsdcg-id>

FreeBSD & Google Summer of Code 2016

The Google Summer of Code is held every year, giving students around the world an opportunity to showcase their coding talents. The following participants have been selected to represent FreeBSD on various projects ranging from security, virtualization, kernel, to cloud. Congrats to the following on their submissions to Google Summer of Code!

bhimanshu Add SCSI passthrough to CTL	CTL is the FreeBSD SCSI target layer. There are various SCSI commands, but it's usually used for block-level access. This project focuses on making CTL capable of providing physical
Suraj Ponugoti Adding SCSI passthrough to CTL	The purpose of this project is to export physical SCSI devices with all their features over SCSI through CTL as an actual SCSI target.
Yuanxun Ethernet Ring Protection Switching	Ethernet Ring Protection Switching (ERPS) provide sub-50ms protection and recovery switching for Ethernet traffic in a ring topology and at the same time ensuring that there are
ghost_rider Grant-Table User-Space Device	A grant-table user-space device will allow user-space applications to map and share grants (Xen way to share memory) among Xen domains.
Alex Teaca HD Audio device model in userspace for bhyve	The bhyve hypervisor does not have any sound card emulation at the moment. This project is proposed by Peter Grehan and aims to implement the High Definition Audio Specification
yuri High-performance P4 software switch using netmap	High-performance P4 software switch using netmap
vmaffione High-performance TCP/IP networking for bhyve VMs using netmap passthrough	Modern cloud computing technologies require powerful virtualization tools. The bhyve hypervisor is currently missing an high performance Virtual Machine networking solution.
Fabian Freyer Improving libvirt support for bhyve	The primary aim of this project would be to improve libvirt support for bhyve. Missing API functions in the libvirt bhyve driver shall be implemented and where necessary interfaces to
Dashhh libnrv improvements	The libnrv library allows to easily manage name value pairs as well as send and receive them over sockets. You are able to add booleans, strings, numbers, descriptors, binaries or even nest
Omp Non BSM to BSM Conversion Tools	Let's imagine a FreeBSD server which collects audit records from machines that are not necessarily using BSM as the format of their audit records. The idea is to create a tool which
jesa Support bhyve as a Vagrant VM backend	Vagrant is a tool to create and manage virtual environment, which can be used to for creating and configure lightweight, reproducible, and portable development environments. This project
Shivansh Rai TCP/IP Regression Test Suite	Regression testing is one of the most critical elements of the test artifacts and proves to be one of the most preventive measures for testing a software. Currently, within FreeBSD, there is

FreeBSD Google Summer of Code 2016:
<https://summerofcode.withgoogle.com/organizations/4892834293350400/>

Past Summer of Codes:
<https://www.freebsd.org/projects/summerofcode.html>

<https://www.freebsdnews.com/2016/09/30/freebsd-google-summer-code/>

iXsystems to Host MeetBSD California 2016 at UC Berkeley



Conference to Return to the “Birthplace of BSD” for Its Fifth Installment

SAN JOSE, CA—(Marketwired – September 06, 2016) – iXsystems announced today that the fifth MeetBSD California conference will take place at UC Berkeley’s Clark Kerr campus on November 11-12. As in past years, this year’s MeetBSD California will once again follow a mixed “unConference” format and will feature breakout sessions, discussion groups, and talks from prominent figures in the BSD community.

MeetBSD California is the premier BSD Conference in the San Francisco Bay Area. Since its inception in 2008, MeetBSD California has been held every two

years in Silicon Valley, bringing together BSD community members from all over the region and around the world.

Previous settings for MeetBSD California have included the Google and Yahoo! Campuses, Hacker Dojo in Mountain View, and the Western Digital campus in San Jose. The BSD operating system was developed in the early '90s at this year’s venue, UC Berkeley.

Kirk McKusick, one of the instigators of BSD at Berkeley in the 1980s, says, “I am thrilled to have a BSD Conference return to the campus at which it started. I look forward to catching up on all the latest work going into the BSD systems and especially look forward to the party at the historic Hillside Club on Friday evening.”

“For this fifth installment of the MeetBSD California conference, we’re proud to bring it home to where it all began,” says Matt Olander, Co-Founder and Chief Science Officer of iXsystems. “UC Berkeley provides the perfect backdrop for the accomplishment of BSD related development milestones. We’re looking forward to the insightful discussions that will take place at this year’s MeetBSD.”

For more information about MeetBSD or to register to attend, visit MeetBSD.com or email info@meetbsd.com.

<https://www.ixsystems.com/blog/ixsystems-host-meetbsd-california-2016-uc-berkeley/>

FreeBSD 11 Released — The Open Source Operating System Gets New Features

FreeBSD Release Engineering Team has announced the general availability of FreeBSD 11 open source operating system. This is the first release of the stable/11 branch and it comes with many security improvements and better hardware support.

Towards the end of the last month, we reported that the final images of FreeBSD 11 have started to appear on the FTP servers before the official release. Now, the official release version of FreeBSD 11 is here and it's available for download.

Security improvements in the final release

The users who have installed the bootleg version a couple of weeks ago, they need to upgrade their systems. Wondering why? It's because the developers have rebuilt and republished the FreeBSD 11 mirrors due to some last minute security fixes.

The announcement mentions fixes to OpenSSL, BIND, Bspatch, Portsnap, and Libarchive. You'll also see OpenSSH security patches along with an upgrade to 7.2p2.

The latest release of this open source operating system comes with new architecture support, toolchain enhancements, performance improvements, and support for the contemporary wireless chipset.

The FreeBSD Foundation says that the new release represents years of hard work by the members of the large FreeBSD community.

Better hardware support in FreeBSD 11

The svn-lite utility has been updated to version 1.9.4. FreeBSD 11 also brings the support for the AArch64 (arm64) architecture and bhyve hypervisor. There's also out-of-the-box support for Raspberry Pi, Raspberry Pi 2, and Beaglebone Black peripherals.

FreeBSD 11 is now available for different architectures including amd64, i386, powerpc, powerpc64, sparc64, armv6, and aarch64. The users can install this open source operating system from bootable ISO images or over the network.

For more information, you can read the release announcement. Get the FreeBSD 11 images here on the download page.

<https://www.freebsdnews.com/2016/09/30/freebsd-11-0-release-delayed/>

OpenBSD Founder Calling For LLVM To Face A Cataclysm Over Its Re-Licensing

```

230 EXPORT_SYMBOL_GPL(cgroup_is_descendant);
231
232 static int cgroup_is_releasable(const struct cgroup *cgrp)
233 {
234     const int bits =
235         (1 << CGRP_RELEASABLE) |
236         (1 << CGRP_NOTIFY_ON_RELEASE);
237     return (cgrp->flags & bits) == bits;
238 }
239
240 static int notify_on_release(const struct cgroup *cgrp)
241 {
242     return test_bit(CGRP_NOTIFY_ON_RELEASE, cgrp->flags);
243 }
244
245 /**
246  * for_each_css - iterate all css's of a cgroup
247  * @cgrp: the cgroup to iterate
248  * @css: the iteration cursor
249  * @fn: the function to call for each css
250  */

```

For over one year, there's been talk of LLVM pursuing a mass relicensing from its University of Illinois/NCSA Open Source License, which is similar to the three-clause BSD license, to the Apache 2.0 license with explicit mention of GPLv2 compatibility. As mentioned in that aforelinked article, this re-licensing is moving ahead. OpenBSD leader Theo de Raadt is predicting this could cause a major problem and is in fact hoping for it.

LLVM/Clang has been popular with many BSD operating systems due to the LLVM/Clang's more liberal licensing. But if they switch to the Apache 2.0 license, Theo de Raadt commented, "I hope a year or two later, some author of a component (especially one from Europe where the moral rights of an author still carries substantial weight) sub-

marines the new license, surfacing to indicate that they never signed off on the additional terms applied to them as a significant author, and will accept no cash to solve the problem. Then they are dead in the water."

If that comes about, then he feels the project could face a cataclysm and that a fork of LLVM/Clang could happen from the last point of the code being under the current license.

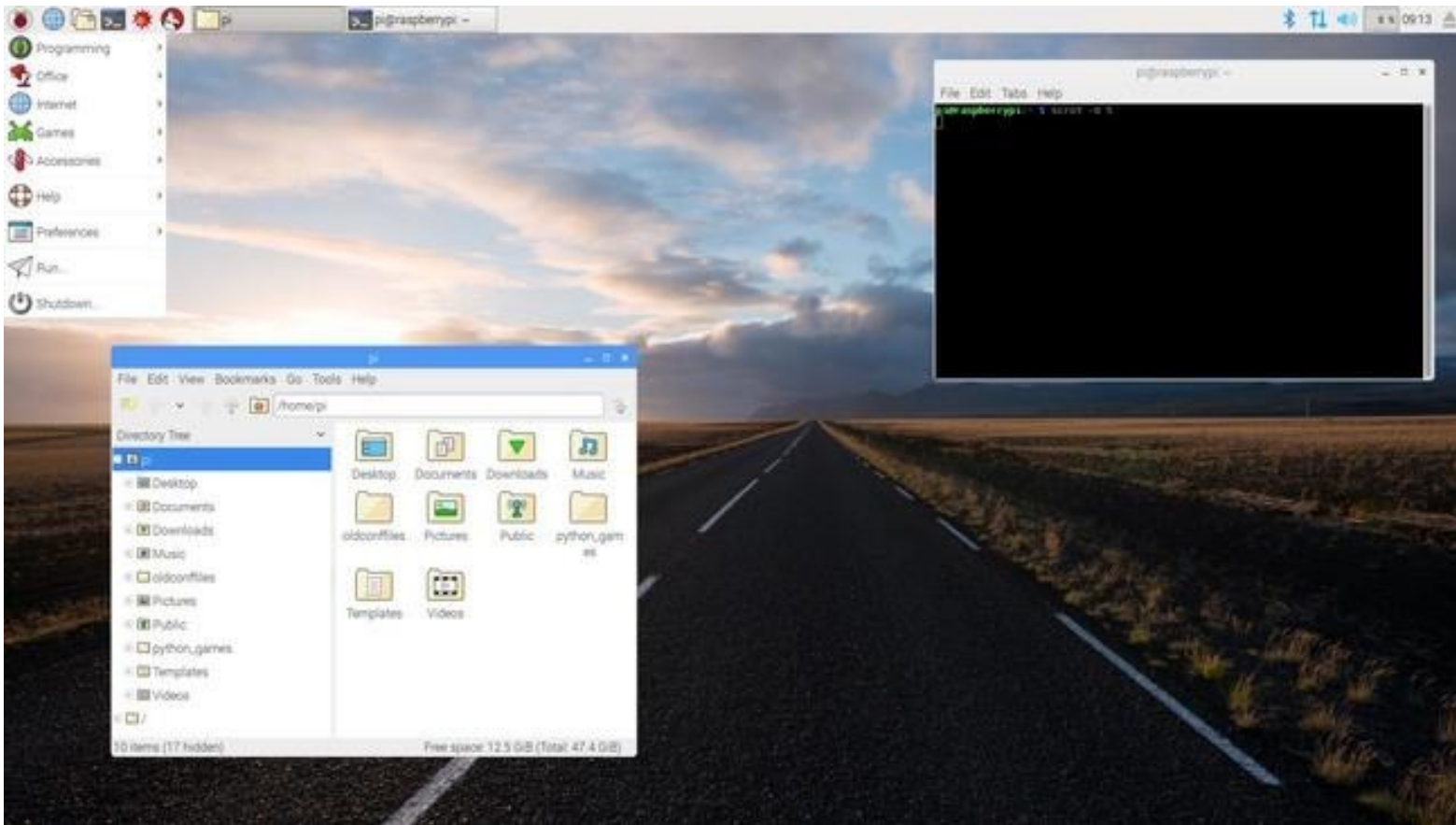
Theo goes as far as calling the current re-licensing push "copyright theft" and "I suspect a few people are being paid a lot of wages to act as agents permitting theft from their co-contributors. They worked with others but now they are ready to steal from them." He's also hoping that someone now will intentionally try to get "a major diff" of code into LLVM now and will ultimately oppose to this re-licensing being pursued by the LLVM Foundation.

http://www.phoronix.com/scan.php?page=news_item&px=LLVM-License-Theo-de-Raadt

Raspberry Pi adds PIXEL eye candy to desktop

Raspbian gets user interface makeover and Chromium browser.

Fruity low-cost computer the Raspberry Pi is constantly getting enhancements, and the latest is an update to its Raspbian Linux build, which has been given a makeover with a new desktop shell called PIXEL and a version of the Chromium browser.

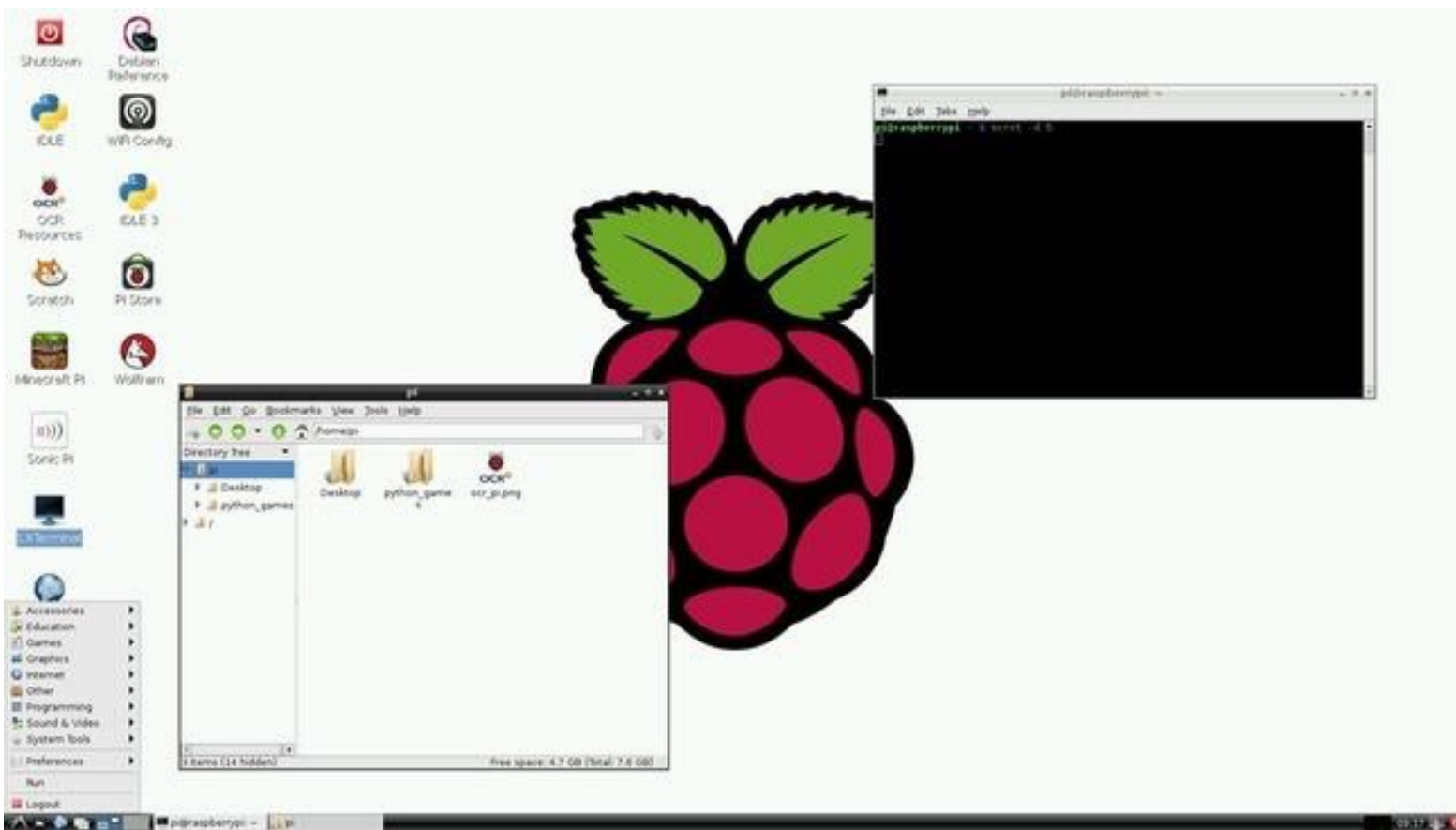


New PIXEL desktop: We know you were missing your nature backgrounds. Pic: RPi Foundation

Raspbian is the default platform that the folks at Raspberry Pi provide for the popular bare board miniature computer. This is based on Debian Linux and has traditionally shipped with a rather spartan desktop user interface known as LXDE.

This has now been superseded by PIXEL, which stands for “Pi Improved Xwindows Environment, Lightweight”. This has apparently been developed from LXDE, but has had so many improvements that it has become “a complete product in its own right and should have its own name”.

PIXEL not only gives the icons on the taskbar, menu and file manager a long overdue makeover, but introduces a choice of background wallpaper images. There is also a new boot-up splash screen that replaces the scrolling Linux startup script messages and handily displays a Raspbian version number.



The original desktop. Screengrab courtesy RPi Foundation.

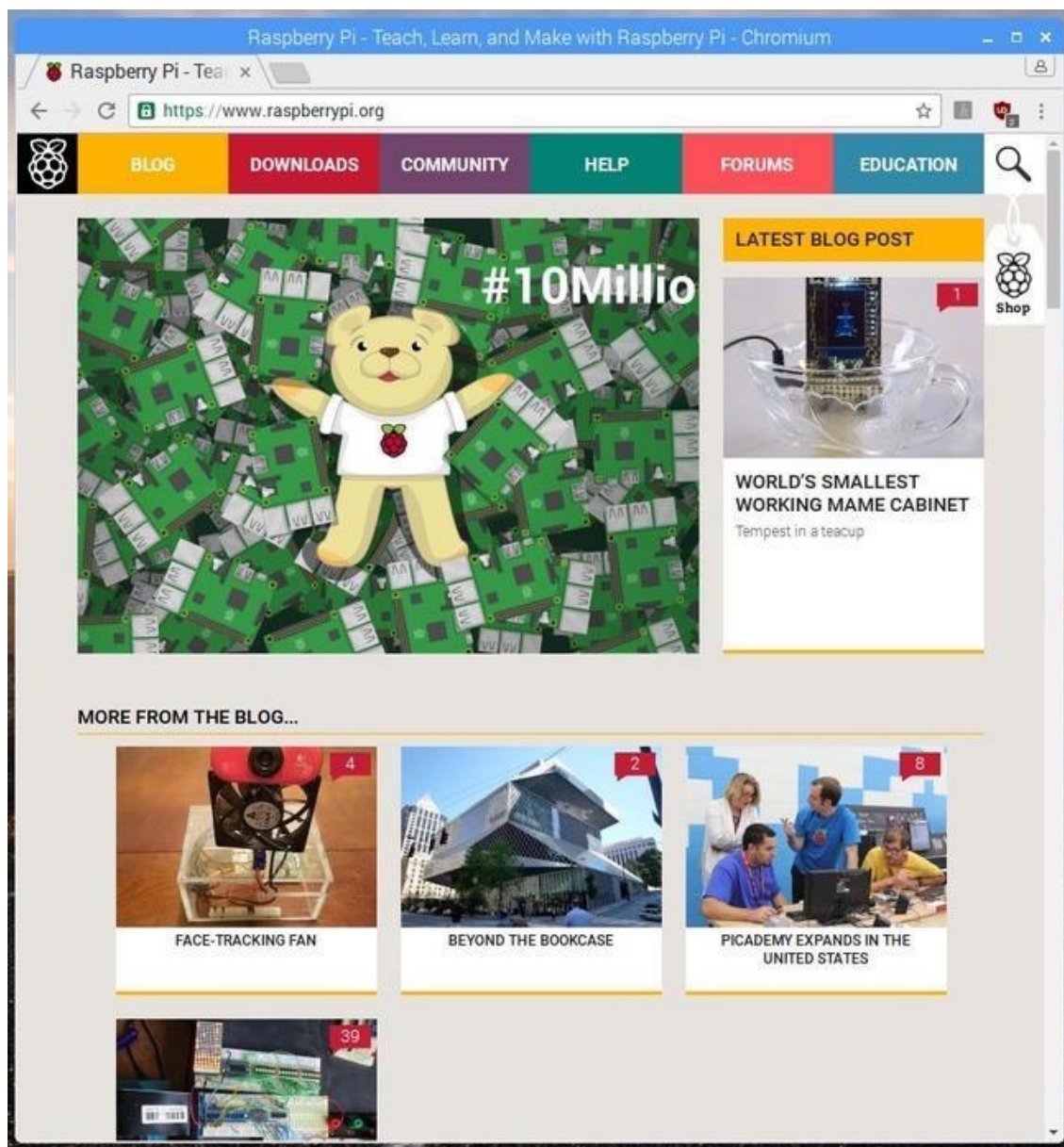
The frame design for individual windows has also been given a more contemporary look, as “the old window design always looked a bit dated compared to what Apple and Microsoft are now shipping,” according to Raspberry Pi’s UX Engineer Simon Long, writing on the Raspberry Pi blog.

For users of the Raspberry Pi 3 device, there are now options in the Wi-Fi and Bluetooth menus to turn off these devices if required. This should also work with most external Wi-Fi and Bluetooth USB dongles.

Browser bling

Perhaps more interesting is an initial release of a version of Chromium for the Raspberry Pi. Chromium is the open source project upon which Google bases its Chrome browser, and is now offered as an alternative to the Epiphany browser that Raspbian has included for the last few years.

With this, Pi users can now enjoy hardware accelerated playback for streamed content, thanks to an included h264ify extension that enables YouTube to serve up H.264 versions of videos. Also included is the uBlock Origin adblocker, purely in the interests of stopping intrusive adverts from slowing down the browsing experience, of course.



Chromium for the Raspberry Pi. Screenshot courtesy RPi Foundation

Another addition is a port of RealVNC's VNC server and viewer applications, enabling users to remote screen into their Raspberry Pi, or alternatively use it as a terminal for controlling other VNC-enabled systems.

These enhancements come at a cost, with Raspberry Pi warning that Chromium in particular is more demanding of hardware resources than the Epiphany browser. While it runs well on the Raspberry Pi 2 and the beefier Raspberry Pi 3, the Pi 1 and Pi Zero hardware may struggle, Long said. ®

http://www.theregister.co.uk/2016/09/28/raspberry_pi_adds_pixel_eye_candy_to_desktop_to_please_users/

Apple to automatically cram macOS Sierra into Macs – 'cos that worked well for Windows 10

And they say Microsoft never innovates anything...

Apple is taking a page from Microsoft's Windows 10 playbook and will push out its latest macOS (ex-OS X) update as an automatic download.

The Cupertino maker of the Performa 275 has confirmed to El Reg that later this week it will begin to push macOS Sierra to Mac owners who have the "automatic update" function enabled on their computers. This feature is usually switched on to receive security fixes and feature updates as soon as possible. Now it'll cause the Sierra upgrade to automatically download onto Macs running OS X El Capitan. Users can install the package by giving the go-ahead in a dialog box.



The move comes less than two weeks after Sierra reached general availability. Up until now, users have had to seek out the update on their own by visiting Apple's Mac Store software service, where it has been posted as a front page download.

Introduced earlier this year at WWDC, Sierra has been touted for its

improved integration with Apple's iOS platform, as well as its borrowing of the Siri personal assistant tool.

To get Sierra, Mac owners will need to be running at least the Late 2009 iMac or MacBook models, the 2010 or newer MacBook Pro, MacBook Air, Mac mini, or Mac Pro. It requires at least 2GB of RAM and 8.8GB of free disk space.

Apple's Sierra rollout looks to be much smoother than Microsoft's disastrous Windows 10 update shove. That effort saw Microsoft roundly criticized for an update campaign that many of its customers had deemed far too pushy and deceptive.

Microsoft enraged users who wanted to keep their machines on Windows 7 and 8.1 by force-feeding them Windows 10; for many folks, the operating system would automatically download in the background and install itself, requiring just a reboot to switch across to the new OS, after months of irritating popups and sneaky dialog boxes.

Apple doesn't appear to be cramming its software as hard as Microsoft – you have to have automatic updates enabled and the storage space to take it, and you have to explicitly agree to the installation – but that's because it doesn't have to; Cupertino's customers are conditioned to be extremely loyal to the brand and take whatever Tim Cook and co hand out, whereas Microsoft has spent decades expertly fostering resentment.

As a result, Apple tends to have a much faster (and smoother) uptake for its OS updates than counterparts in the Windows world.

http://www.theregister.co.uk/2016/10/03/apple_automatic_installs_of_macos_sierra/

Great Specials

On FreeBSD® & PC-BSD® Merchandise

Give us a call & ask about our
SOFTWARE BUNDLES

1.925.240.6652

\$39.95

FreeBSD 9.1 Jewel Case CD Set
or FreeBSD 9.1 DVD

\$29.95

PC-BSD 9.1 DVD

\$49.95

The PC-BSD 9.0 Users Handbook
PC-BSD 9.1 DVD

\$99.95

The FreeBSD CD or DVD Bundle

Inside each CD/DVD Bundle, you'll find:
FreeBSD Handbook, 3rd Edition
Users Guide FreeBSD Handbook, 3rd Edition, Admin Guide
FreeBSD 9.1 CD or DVD set
FreeBSD Toolkit DVD



Stylish Dress Attire
Look Your Professional Best



Comfy Apparel
Stay Warm in Zip Ups & Pullovers

T-Shirts
Lots of Styles to Choose From

FreeBSD 9.1 Jewel Case CD/DVD \$39.95

CD Set Contains:

- Disc 1** Installation Boot LiveCD (i386)
- Disc 2** Essential Packages Xorg (i386)
- Disc 3** Essential Packages, GNOME2 (i386)
- Disc 4** Essential Packages (i386)

FreeBSD 9.0 CD \$39.95

FreeBSD 9.0 DVD \$39.95

FreeBSD Subscriptions

Save time and \$\$\$ by subscribing to regular updates of FreeBSD

FreeBSD Subscription, start with CD 9.1 \$29.95

FreeBSD Subscription, start with DVD 9.1 \$29.95

FreeBSD Subscription, start with CD 9.0 \$29.95

FreeBSD Subscription, start with DVD 9.0 \$29.95

PC-BSD 9.1 DVD (Isotope Edition)

PC-BSD 9.1 DVD \$29.95

PC-BSD Subscription \$19.95

The FreeBSD Handbook

The FreeBSD Handbook, Volume 1 (User Guide) \$39.95

The FreeBSD Handbook, Volume 2 (Admin Guide) \$39.95

The FreeBSD Handbook Specials

The FreeBSD Handbook, Volume 2 (Both Volumes) \$59.95

The FreeBSD Handbook, Both Volumes & FreeBSD 9.1 \$79.95

PC-BSD 9.0 Users Handbook \$24.95

BSD Magazine \$11.99

The FreeBSD Toolkit DVD \$39.95

FreeBSD Mousepad \$10.00

FreeBSD & PCBSD Caps \$20.00

BSD Daemon Horns \$2.00



Bundle Specials!
Save \$\$\$

Just Plain Fun
Mousepads & Novelty Horns



BSD Magazine
Available Monthly



For even MORE items
visit our website today!

www.FreeBSDMall.com

Serverless Approach to Security Automation

by Renan Dias

Everyone is talking about DevOps (Development and Operations). Some people say DevOps is the job of a single man, some other people say DevOps is rather a culture and is about the collaboration between the development team and the operations team. The truth is that, regardless of the conflicting ideas, everyone agrees that one of DevOps' main goals is automation. Software release automation, right? Well, not quite. There are a lot of things that can be automated: software release, infrastructure provisioning, testing, benchmarking and security, to name a few. Now, did you notice the last thing on this list? Security.

It seems that people on the DevOps wave are only focusing on software release and not treating security as a first-class citizen. For that reason, a new term has been shed light on recently: DevSecOps (or DevOpsSec). DevSecOps aims to turn Security into a first-class citizen in the DevOps wave by treating security as code and automating security procedures. For instance, suppose a company deals with sensitive information at the infrastructure level. They probably have procedures in place to test their infrastructure to make sure there are as few security glitches as possible. However, depending on the size of the infrastructure, it might turn out to be quite expensive to run all of these tests manually, which means that these procedures need to be automated. They could automate running vulnerability scans and penetration tests when significant changes have been pushed to the infrastructure, for example. If you had never thought about automating security procedures, that is what this article is all about.

Problem

A tainted server is a system where there has been any sort of unauthorized activity. This means that, if an unauthorized SSH session is open on a server, for instance, this server becomes tainted.

Consider now the following scenario: you have a cluster of servers in Amazon Web Services (AWS) running a web application, and these servers are part of the same Auto Scaling Group (an Auto Scaling Group is a service that scales your cluster up and down depending on the demand). Now, if you had a cluster of 1,000 servers, how would you know, for instance, when an intruder manages to log in to one of the servers? You might have several monitoring tools in place that send a notification to your phone when someone manages to log in to the servers. That is awesome. But the fact is, that depending on when this happens, you may or may not be able to take action. What if it's in the middle of the night after a long day at work? Or what if you are on a highway driving during your holiday? That's one of the reasons why we should bring automation to the InfoSec field.

Solution

In a nutshell, when an unauthorized SSH session is open on a server, a script will kick in and will tag the server as tainted. As soon as the server is tagged, another procedure will decide whether the session seems to be legit or not. If the session is legit, then the server will be untagged. However, if the session is not legit, the server will be shut down.

Technology stack

The solution in this article will use the following technology components:

- Amazon Web Services:
 - EC2
 - Lambda
 - Cloud Watch
 - A W S C o m m a n d L i n e I n t e r f a c e
- Ubuntu 14.04 (instructions for CentOS 7 will also be given)
- Linux-PAM (Pluggable Authentication Modules)
- Python

This is how all of the above will be used to build the solution: you will first launch an EC2 instance and configure PAM to execute a shell script when an SSH session is open. This shell script will then use the AWS Command Line Interface to add a tag with key tainted and value true to the instance. In one of the tests, this EC2 instance will also have another tag with key manageable and value true. The presence of the tag manageable indicates that if an ssh session is established, it will not be seen as an unauthorized access, but rather as an access that was made in order to carry out some sort of maintenance (which the administrator is aware of). After configuring PAM, you will create an AWS Lambda function written in Python, which will be triggered by Cloud Watch. Cloud Watch will be responsible for checking when a new tag is added to an instance and then will trigger the Lambda function. The function, in turn, will get the instance ID and will check whether the tainted and manageable tags are present. If both are, the script will only remove the tainted tag and not shut down the instance. Else, if tainted is present and manageable is not, the instance will be stopped.

Step 1: Launch an EC2 instance and configure PAM

The first step will be to launch an EC2 instance. If you're already an EC2 instance launching expert, just launch an instance with the operating system of your choosing, attach an Identity and Access Management (IAM) role which allows the "ec2:CreateTags" action, add a tag to your instance with the key manageable and value true, and then skip to the part where PAM will be configured. If you have not ever set up an EC2 instance, keep reading.

Go to the AWS console and click on EC2:

The screenshot shows the AWS Management Console interface. At the top, there's a navigation bar with the AWS logo and various service links. Below this, the 'Amazon Web Services' section is displayed, categorized into several groups. The 'Compute' group is highlighted, and within it, the 'EC2' service (Virtual Servers in the Cloud) is selected and outlined with a red box. Other services visible include EC2 Container Service, Elastic Beanstalk, and Lambda. To the right, there's a 'Resource Groups' section with a 'Create a Group' button and a 'Tag Editor' button. Below that, 'Additional Resources' are listed, including links to 'Getting Started', 'AWS Console Mobile App', 'AWS Marketplace', and 'AWS re:Invent Announcements'. At the bottom right, the 'Service Health' section shows a green checkmark indicating that all services are operating normally.

Then, click on Launch Instance:

Resources

You are using the following Amazon EC2 resources in the US West (Oregon) region:

0 Running Instances

0 Elastic IPs

0 Dedicated Hosts

0 Snapshots

0 Volumes

0 Load Balancers

6 Key Pairs

10 Security Groups

0 Placement Groups

Build and run distributed, fault-tolerant applications in the cloud with [Amazon Simple Workflow Service](#).

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Note: Your instances will launch in the US West (Oregon) region

Service Health

Service Status:

US West (Oregon):

Scheduled Events

US West (Oregon):

No events

To launch an EC2 instance with Ubuntu 14.04, click on Ubuntu Server 14.04 LTS (HVM), SSD Vol-
ume Type:

Step 1: Choose an Amazon Machine Image (AMI)

Cancel and Exit

Red Hat

Free tier eligible

Red Hat Enterprise Linux version 7.2 (HVM), EBS General Purpose (SSD) Volume Type

Root device type: ebs Virtualization type: hvm

SUSE Linux

Free tier eligible

SUSE Linux Enterprise Server 12 SP1 (HVM), SSD Volume Type - ami-d2627db3

SUSE Linux Enterprise Server 12 Service Pack 1 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.

Root device type: ebs Virtualization type: hvm

Ubuntu

Free tier eligible

Ubuntu Server 14.04 LTS (HVM), SSD Volume Type - ami-d732f0b7

Ubuntu Server 14.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Root device type: ebs Virtualization type: hvm

Windows

Free tier eligible

Microsoft Windows Server 2012 R2 Base - ami-2827f548

Microsoft Windows 2012 R2 Standard edition with 64-bit architecture. [English]

Root device type: ebs Virtualization type: hvm

Select

Select

Select

Select

To launch an EC2 instance with CentOS 7, click on AWS Marketplace (left-hand side) and type in CentOS 7. The console will show at the top the CentOS 7 AMI - CentOS 7 (x86_64) - with Updates HVM:

AWS

Services

EC2

S3

RDS

ElastiCache

Route 53

IAM

Edit

Renan Santiago Dias

Oregon

Support

1. Choose AMI

2. Choose Instance Type

3. Configure Instance

4. Add Storage

5. Tag Instance

6. Configure Security Group

7. Review

Step 1: Choose an Amazon Machine Image (AMI)

Cancel and Exit

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start

My AMIs

AWS Marketplace

Community AMIs

Categories

All Categories

Software Infrastructure (19)

CentOS 7

CentOS 7 (x86_64) - with Updates HVM

★★★★★ (43) | Sold by Centos.org

\$0.00/hr for software + AWS usage fees

Linux/Unix, CentOS 7 | 64-bit Amazon Machine Image (AMI) | Updated: 2/26/16

This is the Official CentOS 7 x86_64 HVM image that has been built with a minimal profile, suitable for use in HVM instance types only. The image contains just enough ...

More info

Select

From now on, the images will only show the instructions for Ubuntu 14.04, but the equivalent instructions for CentOS 7 will be given when necessary.

After selecting an operating system, choose the size of your instance and click on Next: Configure Instance Details (since PAM does not require a powerful CPU or memory, the t2.nano instance size will do):

AWS

Services

EC2

S3

RDS

ElastiCache

Route 53

IAM

Edit

Renan Santiago Dias

Oregon

Support

1. Choose AMI

2. Choose Instance Type

3. Configure Instance

4. Add Storage

5. Tag Instance

6. Configure Security Group

7. Review

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance types Current generation Show/Hide Columns

Currently selected: t2.nano (Variable ECUs, 1 vCPUs, 2.4 GHz, Intel Xeon Family, 0.5 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
<input checked="" type="checkbox"/>	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.small	1	2	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.medium	2	4	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	t2.large	2	8	EBS only	-	Low to Moderate
<input type="checkbox"/>	General purpose	m4.large	2	8	EBS only	Yes	Moderate

Cancel

Previous

Review and Launch

Next: Configure Instance Details

Feedback

English

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

17

MAGAZINE BSD

Feel free to change any configuration you'd like on this page. The most important thing, though, is the IAM role. The instance will need a role that allows the action `CreateTags`. If you happen to already have a role with such permission, select it using the drop-down list. Else, create a new role by clicking on [Create new IAM role](#) (the link will be open in a new tab or window):

The screenshot shows the 'Step 3: Configure Instance Details' page in the AWS Management Console. The navigation bar at the top includes links for AWS, Services, EC2, S3, RDS, ElastiCache, Route 53, IAM, and a user profile. Below the navigation bar, a progress bar indicates the current step is '3. Configure Instance'. The main content area contains several configuration sections: 'Number of instances' (set to 1), 'Purchasing option' (Request Spot instances), 'Network' (vpc-e2aa1187), 'Subnet' (subnet-9cdc6deb), 'Auto-assign Public IP' (Use subnet setting), 'IAM role' (None), 'Shutdown behavior' (Stop), 'Enable termination protection' (unchecked), 'Monitoring' (unchecked), and 'Tenancy' (Shared). A red rectangle highlights the 'Create new IAM role' link next to the IAM role dropdown. At the bottom, there are buttons for 'Cancel', 'Previous', 'Review and Launch', and 'Next: Add Storage'.

This page will list all IAM roles you already created. But before creating a role, you will need to create a policy yourself because there is no AWS Managed Policy with the required permission (you could select the `AmazonEC2ReadOnlyAccess` policy, but we only need the `CreateTags` action - always bear in mind the Principle of Least Privilege). To create a new policy, click on [Policies](#) on the left panel:

The screenshot shows the AWS IAM console 'Policies' page. The left-hand navigation menu includes links for Dashboard, Search IAM, Details, Groups, Users, Roles, Policies (highlighted with a red rectangle), Identity Providers, Account Settings, and Credential Report. The main content area has a 'Create New Role' button and a 'Role Actions' dropdown. Below this is a search bar and a table header with columns 'Role Name' and 'Creation Time'. The table currently shows 'Showing 0 results' and 'No records found'.

Then, click on Create Policy:

Dashboard

Search IAM

Details

Groups

Users

Roles

Policies

Identity Providers

Account Settings

Credential Report

Encryption Keys

Create Policy

Policy Actions

Filter: Policy Type

Filter

Showing 207 results

	Policy Name	Attached Entities	Creation Time	Edited Time
<input type="checkbox"/>	AdministratorAccess	2	2015-02-06 18:39 UTC+0100	2015-02-06 18:39 UTC+0100
<input type="checkbox"/>	AmazonEC2FullAccess	1	2015-02-06 18:40 UTC+0100	2015-02-06 18:40 UTC+0100
<input type="checkbox"/>	AmazonS3FullAccess	1	2015-02-06 18:40 UTC+0100	2015-02-06 18:40 UTC+0100
<input type="checkbox"/>	PowerUserAccess	1	2015-02-06 18:39 UTC+0100	2015-02-06 18:39 UTC+0100
<input type="checkbox"/>	AmazonAPIGatewayAdministr...	0	2015-07-09 18:34 UTC+0100	2015-07-09 18:34 UTC+0100
<input type="checkbox"/>	AmazonAPIGatewayInvokeFull...	0	2015-07-09 18:36 UTC+0100	2015-07-09 18:36 UTC+0100
<input type="checkbox"/>	AmazonAPIGatewayPushToCl...	0	2015-11-11 23:41 UTC+0100	2015-11-11 23:41 UTC+0100
<input type="checkbox"/>	AmazonAppStreamFullAccess	0	2015-02-06 18:40 UTC+0100	2015-02-06 18:40 UTC+0100
<input type="checkbox"/>	AmazonAppStreamReadOnlyA...	0	2015-02-06 18:40 UTC+0100	2015-02-06 18:40 UTC+0100
<input type="checkbox"/>	AmazonCognitoDeveloperAuth...	0	2015-03-24 17:22 UTC+0100	2015-03-24 17:22 UTC+0100
<input type="checkbox"/>	AmazonCognitoPowerUser	0	2015-03-24 17:14 UTC+0100	2016-06-02 17:57 UTC+0100
<input type="checkbox"/>	AmazonCognitoReadOnly	0	2015-03-24 17:06 UTC+0100	2016-06-02 18:30 UTC+0100
<input type="checkbox"/>	AmazonDMSCloudWatchLogs...	0	2016-01-07 23:44 UTC+0100	2016-01-07 23:44 UTC+0100

Feedback

English

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

To create your own policy, select the bottom most option:

Create Policy

Step 1: Create Policy

Step 2: Set Permissions

Step 3: Review Policy

Create Policy

A policy is a document that formally states one or more permissions. Create a policy by copying an AWS Managed Policy, using the Policy Generator, or typing your own custom policy.

Copy an AWS Managed Policy

Start with an AWS Managed Policy, then customize it to fit your needs.

Select

Policy Generator

Use the policy generator to select services and actions from a list. The policy generator uses your selections to create a policy.













Select

Create Your Own Policy

Use the policy editor to type or paste in your own policy.

Select

Name the policy TaintedServerEC2Access and add a brief description (you could leave the description empty if you'd like):

 **AWS**  **Services**  **EC2**  **S3**  **RDS**  **ElastiCache**  **Route 53**  **IAM**  **Edit**  **Renan Santiago Dias**  **Global**  **Support**

Create Policy

[Step 1: Create Policy](#)

[Step 2: Set Permissions](#)

Step 3: Review Policy

Review Policy

Customize permissions by editing the following policy document. For more information about the access policy language, see [Overview of Policies](#) in the *Using IAM* guide. To test the effects of this policy before applying your changes, use the [IAM Policy Simulator](#).

Policy Name

Description

Policy Document

1

☒ Use autoformatting for policy editing

[Cancel](#) [Validate Policy](#) [Previous](#) [Create Policy](#)

Now, in regards to the policy document, copy and paste the following JSON object:

```
{

  "Version": "2012-10-17",

  "Statement": [{

    "Sid": "Stmt1475523748249",

    "Action": [

      "ec2:CreateTags"

    ],

    "Effect": "Allow",

    "Resource": "*"

  }]

}
```

A Policy Document is a JSON object that states the methods of the AWS API that can or cannot be called by identities (users, groups, and roles). The policy document above, for instance, allows these identities to call the method `CreateTags` of the AWS API for any resource. If you wish to restrict this permission to a specific resource (e.g., an S3 bucket), you will need to get the resource's Amazon Resource Name (ARN). Learn more about Policy Documents* and Amazon Resource Names**.

Click on Validate Policy to make sure there is no syntax error:

The screenshot shows the AWS IAM console interface for creating a new policy. The top navigation bar includes links to various AWS services like EC2, S3, RDS, ElastiCache, Route 53, and IAM. The left sidebar shows the 'Create Policy' workflow with three steps: 'Step 1: Create Policy', 'Step 2: Set Permissions', and 'Step 3: Review Policy'. The main content area displays a policy document for 'TaintedServerEC2Access' with the description 'Allows EC2 instance to tag itself and other instances'. The policy document is a JSON object with the following structure:

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "Stmt1475523748249",
6       "Action": [
7         "ec2:CreateTags"
8       ],
9       "Effect": "Allow",
10      "Resource": "*"
11    }
12  ]
13 }
```

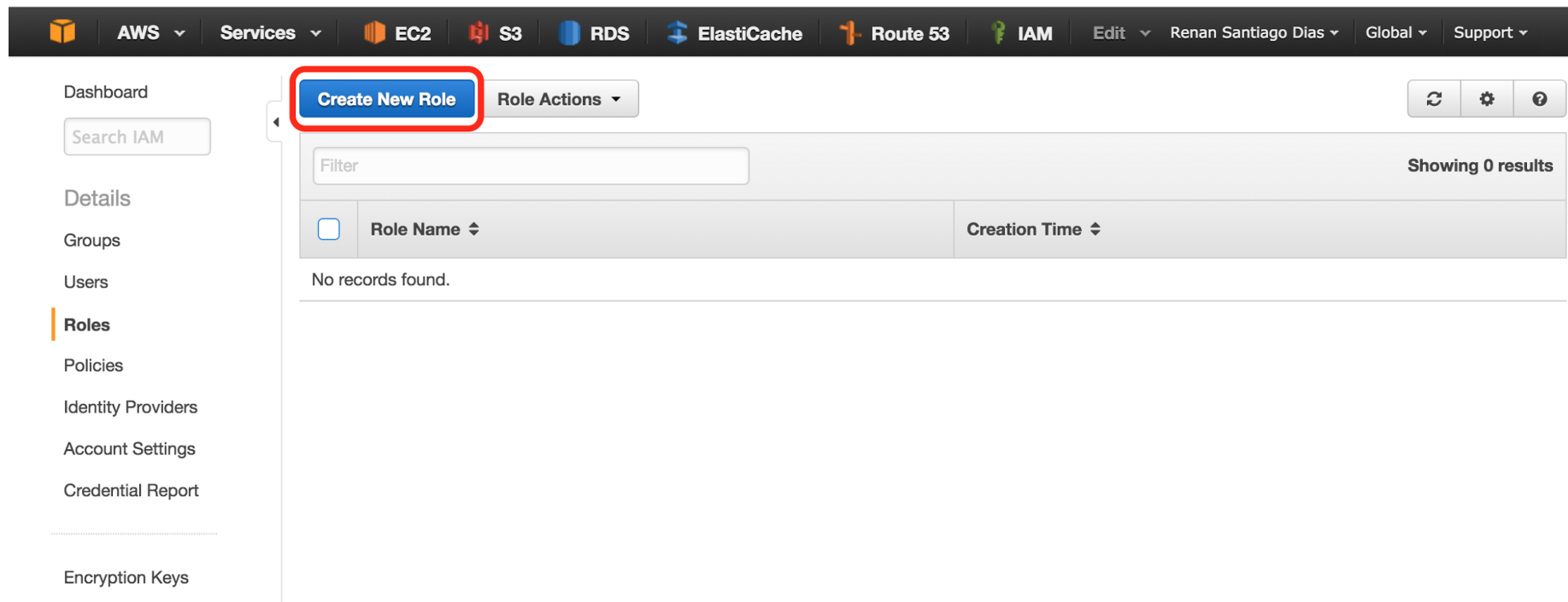
Below the policy document, there is a checkbox labeled 'Use autoformatting for policy editing' which is checked. At the bottom right, there are four buttons: 'Cancel', 'Validate Policy' (highlighted with a red box), 'Previous', and 'Create Policy'.

After successfully validating your policy, click on Create Policy.

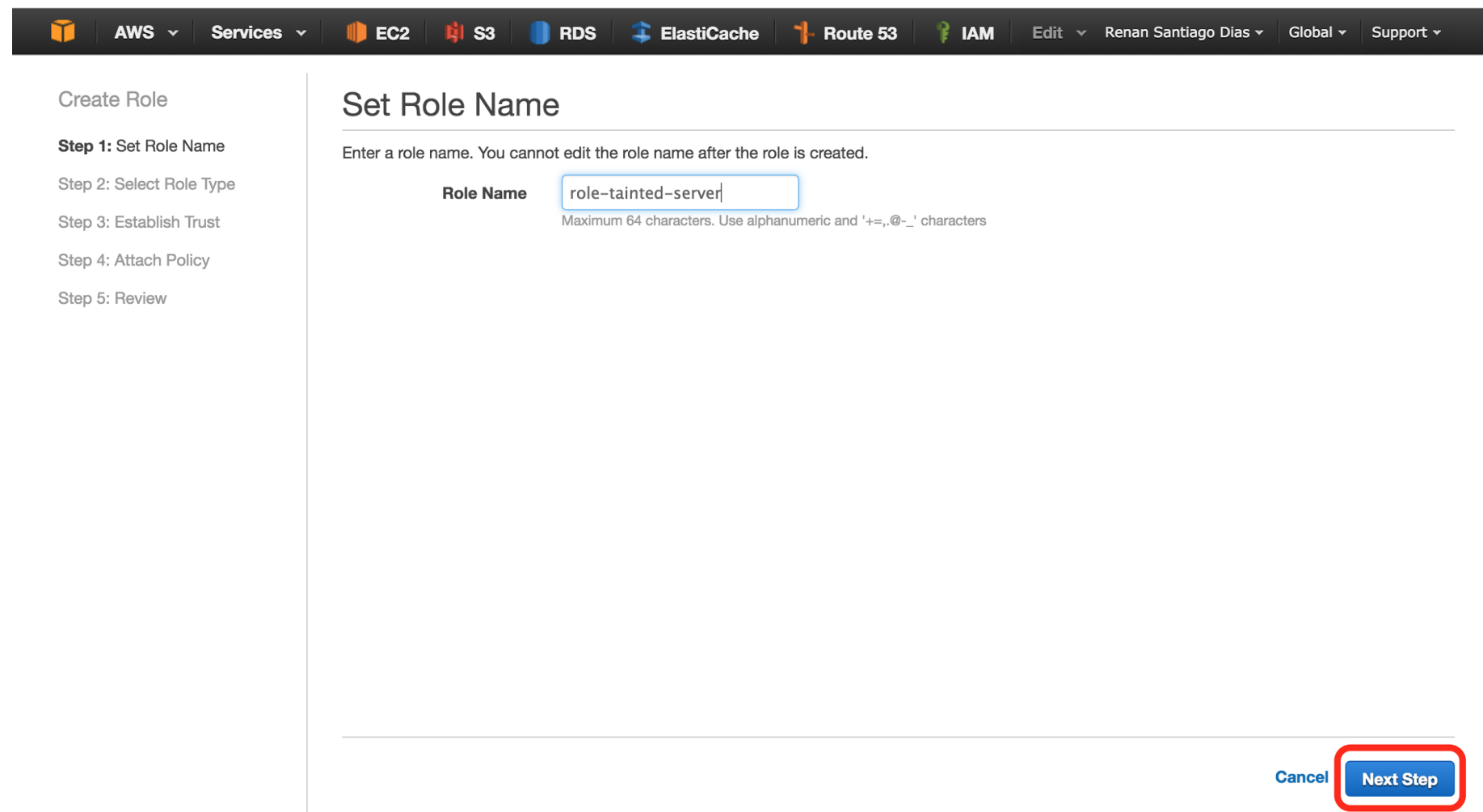
*http://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements.html

**<http://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html>

Now that you have your own policy, create a new role by clicking on Roles on the left panel and then on the Create New Role button at the top:



Call it role-tainted-server and click on Next Step to move forward:



Now select the Amazon EC2 role type:

Create Role

Step 1: Set Role Name

Step 2: Select Role Type

Step 3: Establish Trust

Step 4: Attach Policy

Step 5: Review

Select Role Type

AWS Service Roles

Amazon EC2

Allows EC2 instances to call AWS services on your behalf.

Select

AWS Directory Service

Allows AWS Directory Service to manage access for existing directory users and groups to AWS services.

Select

AWS Lambda

Allows Lambda Function to call AWS services on your behalf.

Select

And type TaintedServerEC2Access to filter the managed policy you've just created. Select it and click Next Step:

Create Role

Step 1: Set Role Name

Step 2: Select Role Type

Step 3: Establish Trust

Step 4: Attach Policy

Step 5: Review

Attach Policy

Select one or more policies to attach. Each role can have up to 10 policies attached.

Filter: Policy Type

TaintedServerEC2Access

Showing 1 results

	Policy Name	Attached Entities	Creation Time	Edited Time
<input checked="" type="checkbox"/>	TaintedServerEC2Access	0	2016-10-03 20:52 UTC+0100	2016-10-03 20:52 UTC+...

Cancel

Previous

Next Step

After creating the role, go back to the EC2 launching page and click on the refresh arrow so the console refreshes the list of roles and shows the role you’ve just created. Select your role and hit Next: Add Storage:

EC2

S3

RDS

ElastiCache

Route 53

IAM

Edit

Renan Santiago Dias

Oregon

Support

1. Choose AMI

2. Choose Instance Type

3. Configure Instance

4. Add Storage

5. Tag Instance

6. Configure Security Group

7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances

1

Launch into Auto Scaling Group

Purchasing option

☐ Request Spot instances

Network

vpc-e2aa1187 (172.31.0.0/16) (default)

Create new VPC

Subnet

subnet-9cdc6deb(172.31.32.0/20) | Default in us-west-2a

Create new subnet

Auto-assign Public IP

Use subnet setting (Enable)

IAM role

None

role-tainted-server

Create new IAM role

Shutdown behavior

Stop

Enable termination protection

☐ Protect against accidental termination

Monitoring

☐ Enable CloudWatch detailed monitoring

Additional charges apply.

Tenancy

Shared - Run a shared hardware instance

Cancel

Previous

Review and Launch

Next: Add Storage

Feedback

English

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

Change the storage configuration if necessary, then hit Next: Tag Instances:

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted
Root	/dev/sda1	snap-47713105	8	General Purpose SSD (GP2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

Cancel

Previous

Review and Launch

Next: Tag Instance

In the Tag Instance page, add two tags:

Key	Value
Name	Tainted Server Test
manageable	true

The Name key is just a suggestion because you will be able to identify this instance more easily in case you have loads of EC2 instances. But the manageable tag is mandatory (later in this article, you will understand why we need it). Add any further tags you find necessary and click on Next: Configure Security Group:

Step 5: Tag Instance
A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. [Learn more](#) about tagging your Amazon EC2 resources.

Key (127 characters maximum)	Value (255 characters maximum)
Name	Tainted Server Test
manageable	true

Create Tag (Up to 50 tags maximum)

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Configure Security Group](#)

On the security group page, select any existing security group that allows inbound traffic on port 22 (SSH) from your IP address (it's a really bad practice to open SSH to 0.0.0.0/0). If you don't have a security group with this rule, create a new security group.

Call it seg-tainted-server, select the My IP option in the drop-down list under Source and hit Review and Launch:

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group
☐ Select an existing security group

Security group name:

Description:

Type <small>i</small>	Protocol <small>i</small>	Port Range <small>i</small>	Source <small>i</small>
SSH <small>⌵</small>	TCP	22	My IP <small>⌵</small> 176.251.184.75/32 <small>✕</small>

Review all the instance launch details and click on Launch. If you already have SSH keys registered in your account, select one of them. Otherwise, create a new key pair, download it and click on Launch Instances:

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click Launch to assign a key pair to your instance and complete the launch process.

AMI Details

Ubuntu Server 14.04 LTS (HVM) - Free tier eligible

Instance Type

Instance Type	ECUs
t2.nano	Variable

Security Groups

seg-tainted-server

Network Performance

Low to Moderate

Key Pair Selection

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair ⌵

Key pair name: tainted-server

You have to download the **private key file** (*.pem file) before you can continue. Store it in a secure and accessible location. You will not be able to download the file again after it's created.

Launch

Grab either the public DNS name or the public IP address of the instance:

Filter by tags and attributes or search by keyword								
1 to 1 of 1								
<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public IP
<input type="checkbox"/>	Tainted Server Test	i-6c939374	t2.nano	us-west-2a	running	Initializing	None	ec2-52-36-141-206.us-west-2.compute.amazonaws.com

Instance: i-6c939374 (Tainted Server Test)		Public DNS: ec2-52-36-141-206.us-west-2.compute.amazonaws.com			
Description		Status Checks		Monitoring	
Tags					
Instance ID	i-6c939374	Public DNS	ec2-52-36-141-206.us-west-2.compute.amazonaws.com		
Instance state	running	Public IP	52.36.141.206		
Instance type	t2.nano	Elastic IPs			
Private DNS	ip-172-31-35-204.us-west-2.compute.internal	Availability zone	us-west-2a		
Private IPs	172.31.35.204	Security groups	seg-tainted-server. view rules		
Secondary private IPs		Scheduled events	No scheduled events		
VPC ID	vpc-e2aa1187	AMI ID	ubuntu/images/hvm-ssd/ubuntu-trusty-14.04-amd64-server-20160714 (ami-		

And ssh into it:

```
$ ssh -i ~/.ssh/tainted-server.pem ubuntu@52.36.141.206
```

```
The authenticity of host '52.36.141.206 (52.36.141.206)' can't be es-
tablished.
```

```
ECDSA key fingerprint is
```

```
SHA256:Zh18mK+6LdMeipbvRJf+q5KhXZq2VaE3Fx+frOOKBbk.
```

```
Are you sure you want to continue connecting (yes/no)? yes
```

```
Warning: Permanently added '52.36.141.206' (ECDSA) to the list of
known hosts.
```

```
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-92-generic x86_64)
```

* Documentation: <https://help.ubuntu.com/>

System information as of Tue Oct 4 17:53:19 UTC 2016

System load: 0.16 Memory usage: 10% Processes:
82
Usage of /: 10.0% of 7.74GB Swap usage: 0% Users logged in:
0

Graph this data and manage this system at:

<https://landscape.canonical.com/>

Get cloud support with Ubuntu Advantage Cloud Guest:

<http://www.ubuntu.com/business/services/cloud>

0 packages can be updated.

0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in `/usr/share/doc/*/copyright`.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by

applicable law.

```
ubuntu@ip-172-31-35-204:~$  
}
```

Note: if you received the message below:

```
Warning: Permanently added '52.36.141.206' (ECDSA) to the list of known hosts.  
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
                @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
      @      WARNING: UNPROTECTED PRIVATE KEY FILE!      @  
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
                @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
Permissions 0644 for '/Users/renandias/.ssh/tainted-server.pem' are too open.  
  
It is required that your private key files are NOT accessible by others.  
  
This private key will be ignored.  
  
Load key "/Users/renandias/.ssh/tainted-server.pem": bad permissions  
  
Permission denied (publickey).
```

Just change the permission of your SSH key to 600:

```
$ chmod 600 <path-to-your-ssh-key>
```

Once you log into the server, open the PAM configuration for the SSH daemon located at `/etc/pam.d/sshd`:

```
ubuntu@ip-172-31-35-204:~$ sudo vim /etc/pam.d/sshd
```

PAM already comes installed in many Linux distributions. But if your system does not have PAM, do a quick research on how to install it on your distribution.

To execute a script upon successful login, you will use PAM's `pam_exec` module. Add the following rule to the bottom of the file:

```
# Executes a script upon successful login.

session optional pam_exec.so /usr/local/bin/tainted
```

Now, create the `/usr/local/bin/tainted` file with the following code:

```
#!/bin/bash

if [[ "$PAM_TYPE" == "open_session" ]]; then

    instance_id=$(curl http://169.254.169.254/latest/meta-data/instance-id)

    aws ec2 create-tags \
        --resources $instance_id \
        --tags Key=tainted,Value=true \
        --region us-west-2

fi
```

First, the script checks the `PAM_TYPE`. PAM passes data to the script via environment variables. With the `PAM_TYPE`, it's possible to identify whether the user has just opened the session, or closed it, for instance. And that's what the if statement below the shebang line is doing. We are not interested in tagging the instance upon session closure, but when the session has been opened. The `curl` command makes a request to the IPv4 Link-Local address `169.254.169.254` to get the instance's ID. Now, let's make a pause here. I can only imagine that you might be asking yourself what random IP address is this. According to the RFC 3927, an IPv4 Link-Local address is an IP address within the `169.254/16` range to communicate with other devices on the same physical (or logical) link. Amazon hasn't published any implementation details about

how this is done, but my assumption is that there is some sort of device on the instance's physical or logical link responsible for dealing with metadata requests. Back to the script, it then uses the AWS CLI to tag the instance with the tainted key (you will need to provide the region code to the AWS CLI, unless you run `aws configure` and manually set the region to be used as default). Note, though, that the AWS CLI does not come installed on the Ubuntu AMI (or the CentOS AMI). Save the script above and install the AWS CLI with the following commands:

```
$ curl "https://s3.amazonaws.com/aws-cli/awscli-bundle.zip" -o  
"awscli-bundle.zip"  
  
$ unzip awscli-bundle.zip # install unzip if necessary  
  
$ sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/  
aws
```

Note: *you will not need any AWS credential since the instance already has an IAM role attached to it.*

Last but not least, change the script's permission so the system can execute it:

```
$ sudo chmod +x /usr/local/bin/tainted
```

Without any further ado, test if the instance will tag itself after we ssh into it. The test is pretty easy: just terminate your SSH session and log in again.

This should trigger the tainted script and the instance should have a new tag with key tainted and value true:

<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public IP
<input type="checkbox"/>	Tainted Server Test	i-6c939374	t2.nano	us-west-2a	running	2/2 checks ...	None	ec2-5

Instance: i-6c939374 (Tainted Server Test)

Public DNS: ec2-52-36-141-206.us-west-2.compute.amazonaws.com

Description

Status Checks

Monitoring

Tags

Add/Edit Tags

Key	Value	
Name	Tainted Server Test	Hide Column
manageable	true	Show Column
tainted	true	Show Column

Perfect, everything seems to be working! In case your instance does not tag itself, run the tainted script manually and check for any errors. After you fix the error(s), remove the tainted tag and confirm that the instance will tag itself when a user logs in with ssh.

Step 2: Create a Lambda function

Now that the server is ready, you need to create a Lambda function that will execute as soon as an unauthorized ssh session is established. In this article, the function will stop the instance. However, if you wish to use this approach to protect your servers in production, you need to assess your environment first.

Suppose you have a service running on multiple machines in an Auto Scaling Group, and you decide that no one should ssh into these servers to carry out any sort of maintenance. In this case, since you have multiple servers and the Auto Scaling Group is configured to spin new instances as instances go down, it is perfectly fine to stop/terminate upon unauthorized ssh session establishment (unless an ssh session is established in all servers of the Auto Scaling Group at the same time, which indicates a serious security glitch). What kind of action the Lambda function will carry out is totally up to you. The goal of this article is to show you how to use a serverless approach to secure your systems.

Even if the servers in a certain Auto Scaling Group are not to be managed, you still might want to carry out some maintenance given the circumstances. In this case, you could indicate to your Lambda function that everything is fine with the “tainted” instance and it should not be stopped/terminated. This could be accomplished by using a different tag. Remember the manageable tag you created for the instance? That’s right, you got it! If the manageable tag is present, this means that an authorized person is logged in to the instance (if an unauthorized person manages to tag the instance with the manageable tag and ssh into it, then that’s another security glitch you might need to look into).

To summarize the idea of what the Lambda function will do:

- The function will get the instance ID of the tainted server.
- If it finds the manageable tag, it will not stop the instance and will remove the tainted tag instead.
- Else, if it does not find the manageable tag, the instance will be stopped.

The function in this article will be written in Python. But it could be easily ported to either Node.js or Java (which are currently the programming languages supported by AWS Lambda). Here is the function:

```
import boto3

import json

ec2 = boto3.client('ec2')

def lambda_handler(event, context):

    # 1

    # Retrieve EC2 instance ID

    instanceId =
event['detail']['requestParameters']['resourcesSet']['items'][0]['resourceId']
```

```
# 2

# Get all the tags of the instance

response = ec2.describe_tags(

    Filters = [

        {

            'Name': 'resource-id',

            'Values': [

                instanceId

            ]

        }

    ]

)

# 3

isManageable = False

# 4

# Finding out if instance is manageable

for tag in response['Tags']:

    if tag['Key'] == 'manageable' and tag['Value'] == 'true':

        isManageable = True

# If the instance is not manageable, the instance will be shut down
```

if isManageable is False:

```
# 5
```

```
try:
```

```
    response = ec2.stop_instances(
```

```
        InstanceIds = [
```

```
            instanceId
```

```
        ]
```

```
    )
```

```
    print(response)
```

```
except Exception as e:
```

```
    print(e)
```

```
    print("Exception thrown when shutting down instance: " +  
instanceId)
```

```
    raise e
```

```
else:
```

```
# 6
```

```
# Removing tainted tag since the instance is manageable
```

```
try:
```

```
    response = ec2.delete_tags(
```

```
        Resources = [
```

```
            instanceId
```

```
        ],
```

```
        Tags = [
            {
                'Key': 'tainted',
                'Value': 'true'
            }
        ]

    )

    print(response)

    except Exception as e:

        print(e)

        print("Exception thrown when removing Tag tainted")

        raise(e)
```

Here's the breakdown of what the function is doing:

- #1 - Retrieves the instance ID of the instance which triggered the Lambda function (Note: the event dictionary is quite complex, so to get all the correct keys, print the dictionary first to understand how it is structured)
- #2 - Describe the tags of the instance. The response is a dictionary with the key Tags
- #3 - Declares a variable called isManageable. This variable will hold False in case the instance does not have the manageable tag and True otherwise.
- #4 - Loops through the list of tags and sets the variable isManageable to True in case the instance has the manageable tag
- #5 - If there's no manageable tag set, the instance is stopped.
- #6 - If there is a manageable tag set, the instance is not stopped, and the tainted tag is removed.

Time to create this Lambda function!

Go to the AWS Lambda dashboard. If you haven't created a function yet, click on Get Started Now:

AWS

Services

EC2

S3

RDS

ElastiCache

Route 53

IAM

Edit

Renan Santiago Dias

Oregon

Support

AWS Lambda

AWS Lambda is a compute service that runs developers' code in response to events and automatically manages the compute resources for them, making it easy to build applications that respond quickly to new information.

Get Started Now

Learn more about AWS Lambda

S3, Dynamo, Kinesis, SNS, CloudTrail, Mobile, SQS, CloudWatch, IAM, SQS

Respond quickly to new information

AWS Lambda runs your code in response to events such as image uploads, in-app activity, website clicks, or outputs from other AWS services.

Run your code without managing infrastructure

AWS Lambda administers the underlying compute resources for you, so you don't have to.

Cost-effective and efficient

AWS Lambda runs your code only when needed, with no unnecessary overhead or cost.

On the Select Blueprint page, scroll down to the bottom and hit Skip:

Select runtime

Filter

<< < Viewing 1-9 of 51 > >>

s3-get-object-python

An Amazon S3 trigger that retrieves metadata for the object that has been updated.

python2.7 · s3

config-rule-change-triggered

An AWS Config rule that is triggered by configuration changes to EC2 instances. Checks instance types.

nodejs · config

dynamodb-process-stream

An Amazon DynamoDB trigger that logs the updates made to a table.

nodejs · dynamodb

microservice-http-endpoint

A simple backend (read/write to DynamoDB) with a RESTful API endpoint using Amazon API Gateway.

nodejs · api-gateway

node-exec

Demonstrates running an external process using the Node.js child_process module.

nodejs

slack-echo-command-python

A function that handles a Slack slash command and echoes the details back to the user.

python2.7 · api-gateway · slack

simple-mobile-backend

A simple mobile backend (read/write to DynamoDB).

nodejs · mobile

ses-notification-nodejs

An Amazon SES notification handler for processing bounces, complaints and deliveries.

ses

kinesis-process-record-python

An Amazon Kinesis stream processor that logs the data being published.

python2.7 · kinesis

Cancel

Skip

Feedback

English

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

37

MAGAZINE BSD

Then, do not configure any trigger yet and hit Next:

The screenshot shows the AWS Lambda console's 'Configure triggers' step. The top navigation bar includes the AWS logo, 'AWS', 'Services', and various service icons (EC2, S3, RDS, ElastiCache, Route 53, IAM). The left sidebar shows the 'New function' wizard with steps: 'Select blueprint', 'Configure triggers' (highlighted), 'Configure function', and 'Review'. The main content area is titled 'Configure triggers' with the instruction 'Configure an optional trigger to automatically invoke your function.' Below this is a visual representation of a trigger (a dashed box) pointing to a Lambda function icon. A 'Remove' button is next to the trigger icon. At the bottom right, there are three buttons: 'Cancel', 'Previous', and 'Next' (which is highlighted with a red rectangle).

Now the important bit. Call this function `RecycleInstance`, give it a brief description of what it does, and select Python 2.7 (or any other version of Python available):

The screenshot shows the AWS Lambda console's 'Configure function' step. The top navigation bar is identical to the previous screenshot. The left sidebar shows the 'New function' wizard with steps: 'Select blueprint', 'Configure triggers', 'Configure function' (highlighted), and 'Review'. The main content area is titled 'Configure function' with the instruction 'A Lambda function consists of the custom code you want to execute. [Learn more](#) about Lambda functions.' Below this are three form fields: 'Name*' with the value 'RecycleInstance', 'Description' with the value 'Recycles tainted instances', and 'Runtime*' with a dropdown menu showing 'Python 2.7'.

Next, copy and paste the function into the Lambda function code frame:

Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other than boto3). If you need custom libraries, you can upload your code and libraries as a .ZIP file.

Code entry type

Edit code inline

```
1 import boto3
2 import json
3
4 ec2 = boto3.client('ec2')
5
6 def lambda_handler(event, context):
7
8     # 1
9     # Retrieve EC2 instance ID
10    instanceId = event['detail']['requestParameters']['resourcesSet']['items'][0]['resourceId']
11
12    # 2
13    # Get all the tags of the instance
14    response = ec2.describe_tags(
15        Filters = [
16            {
17                'Name': 'resource-id',
18                'Values': [
19                    instanceId
20                ]
21            }
22        ]
23    )
24
25    # 3
26    isManageable = False
```








Scrolling down to the Lambda function handler and role, you do not need to change the Handler field because it's already set up with the correct function to be called (lambda_handler is the default name of the function called by the AWS Lambda service):

Lambda function handler and role

Handler*

lambda_function.lambda_handler

If you defined a function called my_awesome_function instead, then the Handler would be called lambda_function.my_awesome_function. As to the role, select the Create a new role option using the drop-down list. A new tab will be open in the browser:

 **AWS** ▾ **Services** ▾  **EC2**  **S3**  **RDS**  **ElastiCache**  **Route 53**  **IAM** **Edit** ▾ Renan Santiago Dias ▾ Global ▾ Support ▾

AWS Lambda requires access to your resources

AWS Lambda uses an IAM role that grants your custom code permissions to access AWS resources it needs.

▼ Hide Details

Role Summary ⓘ

Role Description Lambda execution role permissions

IAM Role Create a new IAM Role ▾

Role Name lambda_basic_execution

▼ Hide Policy Document

[Edit](#)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ]
    }
  ],
}
```

[Don't Allow](#) [Allow](#)

You can leave the Role Name as is. The important thing is the Policy Document. The Policy Document will state all actions and resources your lambda function will have access to. Basically, your lambda function needs access to:

- Create log group
- Create log stream
- Put log events
- Describe EC2 tags
- Stop instances
- Delete tags

The resulting Policy Document will be the following:

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
{
  "Effect": "Allow",
  "Action": [
    "logs:CreateLogGroup",
    "logs:CreateLogStream",
    "logs:PutLogEvents"
  ],
  "Resource": "arn:aws:logs:*:*:*"
},
{
  "Effect": "Allow",
  "Action": [
    "ec2:DescribeTags",
    "ec2:DeleteTags",
    "ec2:StopInstances"
  ],
  "Resource": "*"
}
]
```

Note: this Policy Document states that the lambda function will be able to stop all instances. If you wish the function to only stop certain instances, get their ARN (Amazon Resource Name) and put them into an array for the Resource key.

Double-check all the information and hit Allow:

AWS

Services

EC2

S3

RDS

ElastiCache

Route 53

IAM

Edit

Renan Santiago Dias

Global

Support

AWS Lambda requires access to your resources

AWS Lambda uses an IAM role that grants your custom code permissions to access AWS resources it needs.

Hide Details

Role Summary

Role Description

Lambda execution role permissions

IAM Role

Create a new IAM Role

Role Name

lambda_basic_execution

Hide Policy Document

logs:PutLogEvents

],

"Resource": "arn:aws:logs:*:*:"

{

"Effect": "Allow",

"Action": [

"ec2:DescribeTags",

"ec2>DeleteTags",

"ec2:StopInstances"

Edit

Don't Allow

Allow

After clicking on Allow, the tab will be closed. Go back to the Lambda page, and you will see your new role:

Lambda function handler and role

Handler*

index.handler

Role*

Choose an existing role

Existing role*

lambda_basic_execution

Note: I have experienced an issue when creating a new role. After creating the role, the lambda page would say that there was an error when creating the role. However, reloading the page was enough to make the role appear when selecting the Choose an existing role option. So, if you experience this issue, just reload the page.

Moving on. The Advanced Settings is useful if you want to dedicate more memory and time to your function. In this case, we don't need more than 128 MB of memory and 3 seconds of timeout

42

MAGAZINE

BSD

since the function is extremely simple. Hit Next:

Advanced settings

These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. [Learn more](#) about how Lambda pricing works.

Memory (MB)*

128

Timeout*

0

min

3

sec

All AWS Lambda functions run securely inside a default system-managed VPC. However, you can optionally configure Lambda to access resources, such as databases, within your custom VPC. [Learn more](#) about accessing VPCs within Lambda. **Please ensure your role has appropriate permissions to configure VPC.**

VPC

No VPC

* These fields are required.

Cancel

Previous

Next

Review all the information and click on Create function:

Review

Please review your Lambda function details. You can go back to edit changes for each section. When you are ready, click **Create function** to complete the setup process.

Lambda function

Edit

Name	RecycleInstance
Description	Recycles tainted instances
Runtime	Python 2.7
Handler	lambda_function.lambda_handler
Existing role*	lambda_basic_execution
Memory (MB)	128
Timeout	3
VPC	No VPC

Cancel

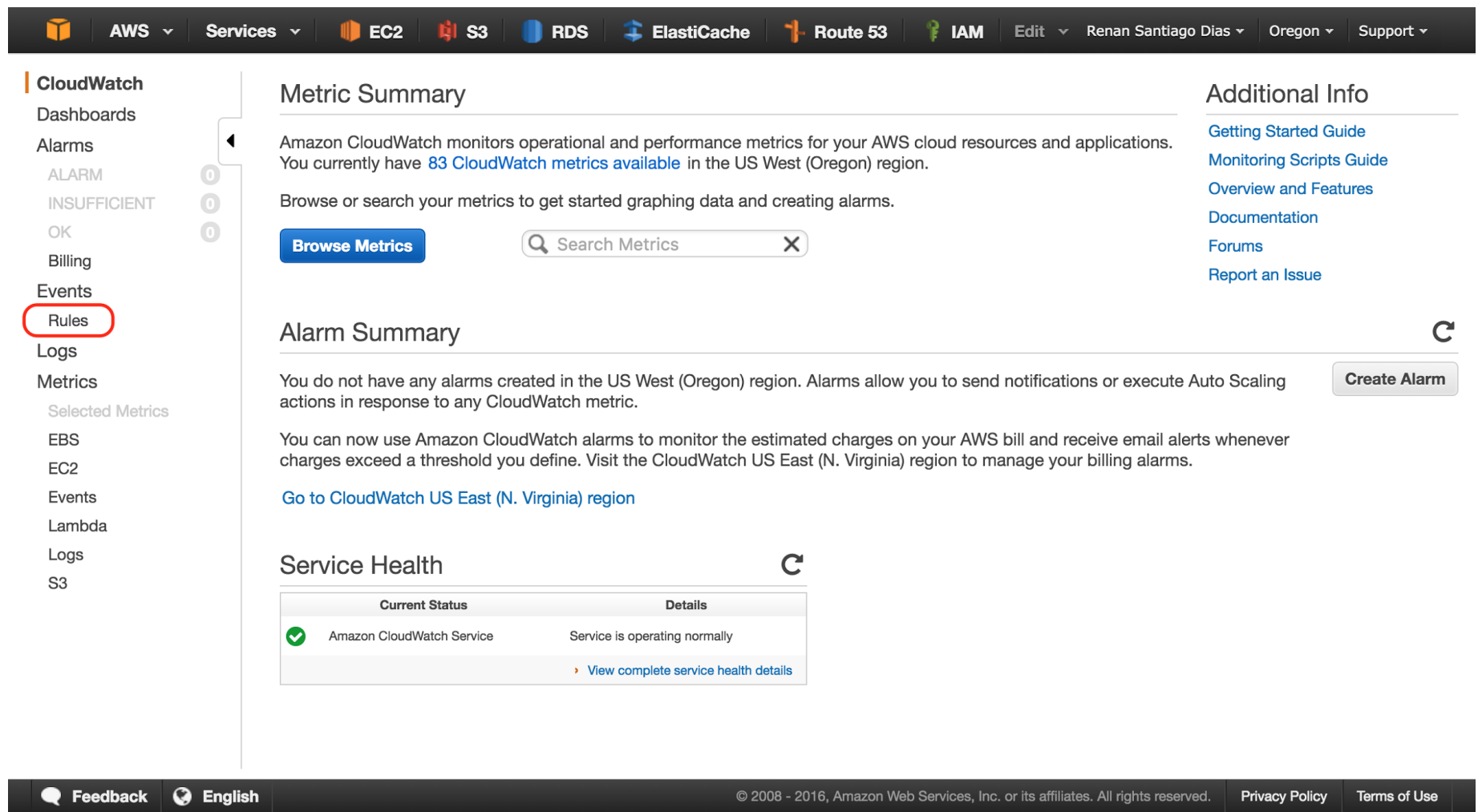
Previous

Create function

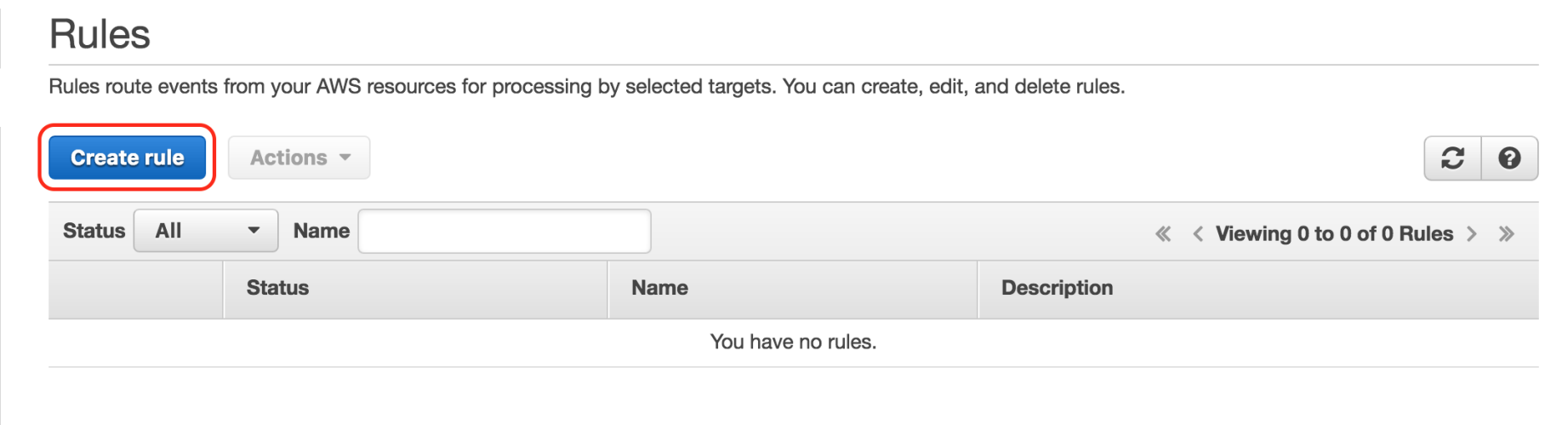
Congratulations! Your (first?) lambda function is ready!

Step 3: Create the Trigger

The third and last step is to create the trigger. To do that, go to the CloudWatch dashboard. On the left-hand side, click on Rules:



Now click on Create rule:



The event source will be the AWS API call. To use the AWS API call as an event, you will need to enable Cloud Trail, which is a service that stores all calls made to the AWS API. If you haven't enabled Cloud Trail yet, do so before proceeding. Now, select EC2 for the Service name.

Finally, click on Specific operation(s) and select CreateTags:

Step 1: Create rule

Create rules to automate actions in your AWS environment.

Event selector

Build a pattern that selects events for processing by your targets.

AWS API call

Service name

EC2

☐ Any operation

☒ Specific operation(s)

×

CreateTags

► Show advanced options

On the right-hand side, you will select the target, which will be the lambda function aforementioned.

Hit Configure details to proceed:

Step 1: Create rule

Create rules to automate actions in your AWS environment.

Event selector

Build a pattern that selects events for processing by your targets.

AWS API call

Service name: EC2

☐ Any operation ☒ Specific operation(s)

CreateTags

[Show advanced options](#)

Targets

Select the targets to receive the events that match the rule you defined.

Lambda function

Function*: RecycleInstance

[Configure version/alias](#)

[Configure input](#)

[Add target*](#)

* Required

[Cancel](#) [Configure details](#)

Choose a name to the rule (I named my rule RecycleInstanceRule), give it a brief description and hit Create rule:

Step 2: Configure rule details

Rule definition

Name*: RecycleInstanceRule

Description: Rule for triggering the RecycleInstance Lambda function

State: ☒ Enabled

CloudWatch Events will add necessary permissions for target(s) so they can be invoked when this rule is triggered.

* Required

[Cancel](#) [Back](#) [Create rule](#)

Phew, looks like everything is in place! Time to test the whole thing now. You will do two tests:

- With the manageable tag
- Without the manageable tag

Let's see what happens in each scenario. First, with the manageable tag. Go to the EC2 dashboard, find the server you created a while ago and add the manageable tag if you haven't done so:

Add/Edit Tags

Apply tags to your resources to help organize and identify them.

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. [Learn more](#) about tagging your Amazon EC2 resources.

Key	Value	
<input type="text" value="Name"/>	<input type="text" value="Tainted Server Test"/>	<input checked="" type="checkbox"/> Hide Column
<input type="text" value="manageable"/>	<input type="text" value="true"/>	<input checked="" type="checkbox"/>

Hit save and you're good to go! Log into your instance.

```
$ ssh -i <your-ssh-key> ubuntu@<ip-address> {
```

As soon as you log in, go back to the AWS Console, refresh the list of instances, and look for the tainted tag:

DescriptionStatus ChecksMonitoringTags

Add/Edit Tags

Key	Value	
Name	Tainted Server Test	Hide Column
manageable	true	Show Column
tainted	true	Show Column

Works like a charm. However, If you have just enabled CloudTrail, wait a few minutes before testing so CloudTrail has enough time to start tracking down API calls. Now, remember about the logic of the lambda function? Since there's a tag with key manageable, the tainted key will be removed. If you refresh the console after a few more seconds, you will notice that the tainted tag is gone:

DescriptionStatus ChecksMonitoringTags

Add/Edit Tags

Key	Value	
Name	Tainted Server Test	Hide Column
manageable	true	Show Column

To make sure there is no mistake, and to confirm your lambda function actually ran, go to the CloudWatch dashboard again. Click on Logs:

CloudWatchDashboardsAlarmsALARMINSUFFICIENTOKBillingEventsRulesLogsMetricsSelected MetricsEBS EC2 Events Lambda Logs S3

CloudWatch > Log Groups

Create Metric FilterActions

Filter: Log Group Name Prefix

Log Groups	Expire Events After	Metric Filters	Subscriptions
<input type="checkbox"/> /aws/lambda/RecycleInstance	Never Expire	0 filters	None

You will notice that there is a log group called /aws/lambda/RecycleInstance. This is the log group that our Lambda function created. Click on it and you will see a log stream:

Search Log GroupCreate Log StreamDelete Log Stream

Filter: Log Stream Name Prefix x

☐ Log Streams

Last Event Time

☐ 2016/10/05/[\$LATEST]b952214f5bac43f38b84d263b9c8478b

2016-10-05 23:01 UTC+1

Whatever you told your function to print, it will be shown there. Now, the second test!

Remove the manageable tag from your instance:

DescriptionStatus ChecksMonitoringTags

Add/Edit Tags

Key	Value	
Name	Tainted Server Test	Hide Column

And ssh into it again. After a few seconds, you will notice two things. The first one is that the instance was tagged with the tainted key as expected. The second is that the instance is being stopped:

AWS

Services

EC2

S3

RDS

ElastiCache

Route 53

IAM

Edit

Renan Santiago Dias

Oregon

Support

EC2 Dashboard

Events

Tags

Reports

Limits

INSTANCES

Instances

Spot Requests

Reserved Instances

Scheduled Instances

Dedicated Hosts

IMAGES

AMIs

Bundle Tasks

ELASTIC BLOCK STORE

Volumes

Snapshots

NETWORK & SECURITY

Security Groups

Elastic IPs

Placement Groups

Key Pairs

Network Interfaces

Launch InstanceConnectActions

Filter by tags and attributes or search by keyword

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public IP
Tainted Server Test	i-6c939374	t2.nano	us-west-2a	stopping		None	ec2-54-148-71-154.us-west-2.compute.amazonaws.com

Instance: i-6c939374 (Tainted Server Test)

Public DNS: ec2-54-148-71-154.us-west-2.compute.amazonaws.com

Description

Status Checks

Monitoring

Tags

Add/Edit Tags

Key	Value	
Name	Tainted Server Test	Hide Column
tainted	true	Show Column

Feedback

English

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

There you go, both tests have passed! You are now officially credentialed to go serverless in order to protect your infrastructure when servers are tainted! But don't think that's the end of it. There is so much more you can do. You could, for example, set up a software called Snort to detect intrusion on your system and notify the personnel responsible for the system via Slack. Or use OSSEC to monitor when a file is altered and log to papertrail. Or use Lynis to identify potential vulnerabilities in your system. Play around with all the mentioned tools in this article a little bit more, and you will realize that automation can not only be easily used to deploy software, but also to protect your infrastructure and systems.



About the Author:

Renan Dias is a passionate DevOps engineer that enjoys learning and writing about Amazon Web Services, infrastructure engineering and automation, HADR at scale, information security, continuous integration and deployment, and distributed systems. In the past, he worked with a wide range of technologies, such as AngularJS, Node.js, MongoDB, Meteor.js, Ionic, HTML/CSS, LAMP stack, iOS SDK, OpenCV, Java and MATLAB. He has also contributed to a Brazilian university's research about Computer Vision and received numerous awards during

his student life.

When not doing any infrastructure-related work, Renan spends his time traveling with his loved ones, learning languages (English, Spanish, German and Polish), practicing sports (Tennis, Ping Pong, Basketball, Cycling, Kung Fu) and playing the electric guitar and the drums.

You can find Renan on LinkedIn: <https://www.linkedin.com/in/renan-dias-a2801163>

Loading an OpenSSH Hostkey From a Hardware Token on FreeBSD

by Mike Tanca

I had a requirement for creating an sftp server that needs strong client and host authentication. The host needs to know it's an authorized client connection, and the client needs to know it's really the host it's connecting to. SSH and public key crypto is great for this, but what if someone steals a copy of your private key? What if someone breaks into your host and makes off with your hostkey? Until you detect the compromise and revoke and regenerate keys, you run the risk of a man in the middle attack, among other things.

One way to mitigate this risk is by keeping your private keys on a hardware token on both sides.

Test setup: FreeBSD RELENG 10, Aladdin eToken 64k (old Style with pkcs15 support). From the ports, OpenCT, OpenSC. I built them from the ports as I wanted OpenSC to use OpenCT as the driver to interact with the Safenet eToken.

Let's start by erasing the token and setting up a pkcs15 filesystem. Note, you might need to initialize the eToken on a Windows box to start from scratch. Don't use these PINs in production. They are there just as an example!:

```
0{sftp}# pkcs15-init -E
Using reader with a card: Aladdin eToken PRO 64k

0{sftp}# pkcs15-init -C -P --pin 12345678 --puk 999999 -a 01 --label
```

```
"server1" --so-pin 12345678 --so-puk 999999 -T
```

Using reader with a card: Aladdin eToken PRO 64k

```
0{sftp}#
```

Now, let's generate the actual private and public key on the token itself. There are two ways you can do this. You can either generate the key off the token and then import it, or you can ask the token to generate it on its own hardware. I think there are caveats to both approaches. If your token dies a hardware death, or let's say a malicious employee or hacker decides to lock the token by too many bad guesses, you are SOL and will need to generate a new key, and then have the entailing fallout from that. Also, how good is the crypto on the token? Everyone loves to beat up OpenSSL, but it is well vetted, and the RND in the *BSD world is very well vetted and understood. Can the same be said for the software on the token? I am not sure either way.

```
0{sftp}# pkcs15-init -G rsa/2048 -a 01 --pin 12345678 --so-pin  
12345678 -u decrypt,sign
```

Using reader with a card: Aladdin eToken PRO 64k

```
0{sftp}#
```

We now have an RSA pair of keys on the token— private and public. Let's read the actual public key in an ssh friendly format.

```
0{sftp}# pkcs15-tool -k
```

Using reader with a card: Aladdin eToken PRO 64k

```
Private RSA Key [Private Key]
```

```
    Object Flags      : [0x3], private, modifiable  
    Usage             : [0x2E], decrypt, sign, signRecover, unwrap  
    Access Flags      : [0x1D], sensitive, alwaysSensitive, neverEx-  
tract, local  
    ModLength         : 2048  
    Key ref           : 16 (0x10)  
    Native            : yes  
    Path              : 3f005015  
    Auth ID          : 01  
    ID                : b146eef6387d12dd3431c758666e18785235bb7b  
    MD:guid           : {cf3b339c-1ad9-b7f4-75a8-c530137c8751}
```

```
:cmap flags : 0x0
:sign       : 0
:key-exchange: 0
```

```
0{sftp}# pkcs15-tool --read-ssh-key
b146eef6387d12dd3431c758666e18785235bb7b
```

Using reader with a card: Aladdin eToken PRO 64k

```
ssh-rsa
AAAAB3NzaC1yc2EAAAFAAL4a91UAAAEBAId3Qzp2kfa8CEcP7x4ooCPw99szSfJIT6MnR
NYLK2KUP/TTuMY6qi6Y2KKSaKyDHPJj6BDPLQ4i+z535+N+iZ/9Vw9sJv70brmBGkNLq2
CsRBENCJeMVapcG5hbCrnVsn/GiEgdSZzF9mxC4o9v+d2ScbEwKsr1X5FDCcMyWUrwM3i
oggQHK4eqB3Wv0WBFo8oNYHqymXiGs5WQ9bF4Mlvpvwbk2mzQUbEtX1xaCK2ehpgtpfTy
QVTfVTKfh+eAPGZSmO6DnpITFht3EE2JLw/Ar+7ERXmbHToG1A7/cIMhGMfdVaTvgnWbt
nTA74cnqojddNVGrZoGS5I9VmR/5a0=
0{sftp}#
```

Let's now use that key for the server. To setup our sftp server, I will create a separate instance listening on port 26. We use the stock OpenSSH config for now. We copy over all the default configs as well as the pre-existing ssh keys. Make the following changes to the config you copied over:

```
0{sftp}# cp -pR /etc/ssh /etc/ssh-26
0{sftp}# diff -u ../ssh/sshd_config sshd_config
--- ../ssh/sshd_config 2015-05-15 18:32:43.945683898 -0400
+++ sshd_config 2015-05-20 09:25:30.834911943 -0400
@@ -14,7 +14,7 @@
 # Note that some of FreeBSD's defaults differ from OpenBSD's, and
 # FreeBSD has a few additional options.

-#Port 22
+Port 26
 #AddressFamily any
```

```
#ListenAddress 0.0.0.0
#ListenAddress ::
@@ -25,10 +25,11 @@
# HostKey for protocol version 1
#HostKey /etc/ssh/ssh_host_key
# HostKeys for protocol version 2
-#HostKey /etc/ssh/ssh_host_rsa_key
+HostKey /etc/ssh-26/ssh_host_rsa-from-agent.pub
#HostKey /etc/ssh/ssh_host_dsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key
+HostKeyAgent /root/etoken-agent

# Lifetime and size of ephemeral version 1 server key
#KeyRegenerationInterval 1h
@@ -55,6 +56,7 @@

# The default is to check both .ssh/authorized_keys and .ssh/
authorized_keys2
#AuthorizedKeysFile .ssh/authorized_keys .ssh/authorized_keys2
+AuthorizedKeysFile /etc/ssh-26/authorized_keys/%u

#AuthorizedPrincipalsFile none
@@ -146,3 +148,10 @@
#      AllowTcpForwarding no
#      PermitTTY no
#      ForceCommand cvs server
+
+Match Group sftponly
+  ChrootDirectory %h
+  ForceCommand internal-sftp
```

```
+ AllowTcpForwarding no
+ PermitTunnel no
+ X11Forwarding no
0{sftp}#
```

Let's now create the `public_key` for the server:

```
0{sftp}# pkcs15-tool --read-ssh-key
b146eef6387d12dd3431c758666e18785235bb7b >
/etc/ssh-26/ssh_host_rsa-from-agent.pub
Using reader with a card: Aladdin eToken PRO 64k
0{sftp}# chmod 600 /etc/ssh-26/ssh_host_rsa-from-agent.pub
```

When the server sees that it's just the public key and not the private key, the daemon will look to the defined agent socket to do all the necessary private key transformations. So pick your socket location in a place on your server that only root has access to.

Next, we fire up the agent with the socket that the server expects to communicate with. We then add to the agent via the `pkcs#11` interface, the path that will let the private key do its magic on the token.

```
0{sftp}# ssh-agent -a /root/etoken-agent
setenv SSH_AUTH_SOCKET /root/etoken-agent;
setenv SSH_AGENT_PID 25563;
echo Agent pid 25563;
0{sftp}# setenv SSH_AUTH_SOCKET /root/etoken-agent;
0{sftp}# ssh-add -s /usr/local/lib/opensc-pkcs11.so
Enter passphrase for PKCS#11:
Card added: /usr/local/lib/opensc-pkcs11.so
0{sftp}#
```

We are now ready to start up the server. Initially, try and do it via debug mode.

```
0{sftp}# /usr/sbin/sshd -d -f /etc/ssh-26/sshd_config
debug1: HPN Buffer Size: 65536
debug1: sshd version OpenSSH_6.6.1p1_hpn13v11 FreeBSD-20140420,
OpenSSL 1.0.1m-freebsd 19 Mar 2015
debug1: key_parse_private2: missing begin marker
debug1: key_parse_private_pem: PEM_read_PrivateKey failed
debug1: read PEM private key done: type
debug1: will rely on agent for hostkey
/etc/ssh-26/ssh_host_rsa-from-agent.pub
debug1: private host key: #0 type 1 RSA
debug1: rexec_argv[0]='/usr/sbin/sshd'
debug1: rexec_argv[1]='-d'
debug1: rexec_argv[2]='-f'
debug1: rexec_argv[3]='/etc/ssh-26/sshd_config'
debug1: Bind to port 26 on ::.
debug1: Server TCP RWIN socket size: 65536
debug1: HPN Buffer Size: 65536
Server listening on :: port 26.
debug1: Bind to port 26 on 0.0.0.0.
debug1: Server TCP RWIN socket size: 65536
debug1: HPN Buffer Size: 65536
Server listening on 0.0.0.0 port 26.
```

In another session, let's just do a keyscan to see what the server serves up and see that it indeed matches the public key that we know.

```
% ssh-keyscan -t rsa -p 26 localhost
# localhost SSH-2.0-OpenSSH_6.6.1_hpn13v11 FreeBSD-20140420
localhost ssh-rsa
AAAAB3NzaC1yc2EAAAFAI4a91UAAAEBAId3Qzp2kfa8CEcP7x4ooCPw99szSfJIT6MnR
NYLK2KUP/TTuMY6qi6Y2KKSaKyDHPJj6BDPLQ4i+z535+N+iZ/
```

```
9Vw9sJv70brmBGkNLq2CsRBENCJeMVapcG5hbCrnVsn/GiEgdSZzF9mxC4o9v+d2ScbEw
Ksr1X5FDCcMyWUrwm3iogqQHK4eqB3Wv0WBFo8oNYHqymXiGs5WQ9bF4Mlvpvwbk2mzQU
bEtX1xaCK2ehpgtpfTyQVTfVTKfh+eAPGZSmO6DnpITFHt3EE2JLw/Ar+7ERXmbHToG1A
7/cIMhGMfdVaTvgnWbtnTA74cnqojddNVGrZoGS5I9VmR/5a0=
```

Let's create the user now to ssh in. In production, do the same pkcs15 key generation on the client's hardware token. But for this example, we will use a traditional ssh key file.

```
0{sftp}# pw groupadd sftponly
0{sftp}# pw useradd testuser1 -g sftponly -m
0{sftp}# chown root /home/testuser1
0{sftp}# mkdir /home/testuser1/files
0{sftp}# chown testuser1 /home/testuser1/files
0{sftp}# chflags schg /home/testuser1
0{sftp}#
```

We create the user and add them to the sftp only group. We ask the user for their public key, and we place it in the directory `/etc/ssh-26/authorized_keys` directory.

```
$ sftp -P 26 192.168.1.1
The authenticity of host ' [192.168.1.1]:26 ([192.168.1.1]:26) ' can't be estab-
lished.
RSA key fingerprint is 58:59:a9:09:3c:c5:92:91:60:dc:d9:f5:0d:d7:92:95.
```

No matching host key fingerprint found in DNS.

Are you sure you want to continue connecting (yes/no)? **yes**

Warning: Permanently added '[192.168.1.1]:26' (RSA) to the list of known hosts.

Enter passphrase for key:

```
' /home/testuser1/.ssh/id_rsa':
```

Connected to 192.168.1.1.

```
sftp> dir
files
sftp> pwd
Remote working directory: /
sftp>
```

On the server, we check:

```
0{sftp}# ssh-keygen -lf ssh_host_rsa-from-agent.pub
2048 58:59:a9:09:3c:c5:92:91:60:dc:d9:f5:0d:d7:92:95
ssh_host_rsa-from-agent.pub (RSA)
0{sftp}#
```

About the Author:

I oversee all things technical at Sentex Communications. My areas of interest are all things IP as in IPv4 and IPv6. Google is probably the best way to see what I am up to. These days I am interested in security as it relates to PCI, IDS/IPS, large scale logging and analysis etc etc.

Specialties: Getting a happy meal out of a stone.... (old joke about making the best with what you have) Article Source:

<http://www.tanrsa.com/mdtblog/?p=73>

Installing Windows 10 using VNC on FreeBSD 11 and Above

by Trent Thompson

This October of 2016 will be a special month for FreeBSD virtualization. Not only will the most recent release of FreeBSD be ready, but it will have been a year since UEFI booting in bhyve was announced via the FreeBSD-Virtualization Mailing List. At the time, bhyve did not have the ability to allow for any type of graphical console, outside of something run on the guest OS like RDP, VNC, or SPICE. Instead, bhyve used a serial console as a means to communicate with the guest operating system.

Most UNIX operating systems these days have the ability to have a VT100-like console. If you have ever had to console into a Cisco Switch or UPS Battery using puTTY, you've probably encountered something similar. Windows has a similar console called Emergency Management Services that allows you to do various administration tasks like change networking and the ability to run CMD.EXE over the serial console. This EMS feature comes standard on official Windows Server Edition installation discs. It is not available by default, and must be enabled by using Windows Unattended XML file baked into the installation disc. At first glance this is a tedious process, but with the help of scripting, it can be easy to accomplish over and over again. If you want to enable the EMS on a regular Windows Desktop OS, it even gets trickier as you need to copy the EMS files from a Server Edition disc over to a Desktop Edition.

This changed in the Spring of 2016 with official support for UEFI-GOP, which allows bhyve to attach a graphical console to the guest OS. This means you can now install Windows the old fashioned way by clicking "Next" a bunch of times. Since these changes made it to the FreeBSD HEAD branch before the FreeBSD 11 release process began, these features are a part of FreeBSD 11 RELEASE. For this tutorial, we are going to assume that you are running at least

FreeBSD 11 RELEASE or newer. If you are using a FreeBSD derivative like TrueOS you should be fine as long as it is based off of FreeBSD 11 or newer. I will be doing this on a machine tracking the 12 CURRENT HEAD branch, but running FreeBSD 11 should work just fine. This tutorial is split into three sections:

- What bhyve is and how to set up your host operating system.
- Obtaining your guest operating system and preparing for installation
- Installing and using Windows 10

If you want to follow along at home, you should already know a little bit about UNIX like operating systems, like Solaris or ones that use the Linux Kernel. In theory, if you have used a Linux Distribution in the past, you should be able to install FreeBSD and follow this tutorial.

Preparing the Host Operating System

The FreeBSD bhyve Hypervisor first appeared in FreeBSD 10 and has grown quite extensively since then. It is a relatively new hypervisor, being younger than Xen, Linux KVM, esxi, VirtualBox, and others. The bhyve hypervisor was also ported to Macintosh OS X as xhyve which is now used by Docker on Mac. The bhyve hypervisor consists of two main components: the VMM kernel module and the bhyve userland utility. There are other userland utilities like grub2-bhyve and bhyveload that use libvmmapi, but we won't go over these in this tutorial.

Before preparing your host, I suggest you read the FreeBSD handbook entry regarding bhyve [here](#). Pay special attention to the sections regarding CPU compatibility and Section 21.7.1 Preparing the Host. The bhyve hypervisor only works on certain models of CPUs, so be sure to check `/var/run/dmesg.boot` to make sure your CPU will run bhyve virtual machines. Once you have determined that your hardware can handle bhyve, we can start to prepare the host operating system's kernel and network configuration. As outlined in the handbook, we need to load the VMM kernel module first. You can do this by simply running `kldload vmm` with super user credentials, or by editing your `/boot/loader.conf` as to avoid having to load the VMM kernel module manually every time you reboot your host. Next we need to set up the networking on the host so the virtual machine can reach the internet. Since we are not going to set up any firewalling or NAT, the virtual machine will appear to be on the same network as the host in this situation.

To do this, we are going to create a network tap device, create a network bridge, then attach the our network interface and tap to the bridge. This sounds complicated, but if you take it a step at a time, it's easier to understand. Before we begin, we need to see what our primary network interface is by running `ifconfig`. From my output, I can see that `igb0` is my primary network interface, as it has an IP address. Your network interface may be something like `em0` or something similar. If you are using WiFi, that is a bit more complicated, and involves creating a NAT and forwarding

traffic around. For the purposes of this tutorial, we won't be going into that. Instead we will deal with physical ethernet connections. Now that we know our interface, we can start by creating the network tap. I have multiple taps created already, so I am going to choose a higher number than 0 and number my tap tap42. You can choose any number you like, as long as it is not already in use. Only one virtual machine can utilize a network tap at a time. To create it, I run `ifconfig tap42 create`. You should now see tap42 in your `ifconfig` output.

Next, we need to tell the kernel to bring a tap device up when it is opened. This is so we don't have to run `ifconfig tap42 up` every time we start the virtual machine. We do this by running `sysctl net.link.tap.up_on_open=1` with super user credentials. You can also set this in `/etc/sysctl.conf` so the change stays after reboot. Now we need to create our bridge. I am going to create a new bridge0. You can create any bridge number you like. We create the bridge with `ifconfig bridge0 create`, and add our devices with `ifconfig bridge0 addm igb0 addm tap42`. Note the use of `igb0`, this may be different depending on what your interface is. To finish things up, we bring the bridge interface up with `ifconfig bridge0 up`. You can refer to the handbook page on how to edit your `/etc/rc.conf` to make these network changes persist after rebooting.

I have simplified this process by writing simple scripts to do the heavy lifting for me. If you take a look my GitHub repo `YetAnotherBhyveScript` or `yabs` here you can find the `hostprep.sh` script to create the bridge and enable the VMM kernel module. Be sure to edit the file before running so it matches your desired network configuration. Once `hostprep.sh` is run, you can setup the network for your guest by attaching it to the bridge you set up using `prepyabs.sh`. The outputs of `hostprep.sh` and `prepyabs.sh` are below:

```
#!/bin/sh

# Prepare host for running bhyve (v0.2)

iface=igb0

bridge=bridge0

# Load kernel module

kldload vmm
```

```
kldload if_tap

kldload if_bridge


# Set sysctl

sysctl net.link.tap.up_on_open=1


# Create bridge

ifconfig ${bridge} create


# Attach interfaces

ifconfig ${bridge} addm ${iface}


# Bring up bridge

ifconfig ${bridge} up


#!/bin/sh


# Prepare host for running bhyve guest (v0.2)


tap=tap42

bridge=bridge0


# Create the tap interface
```

```
ifconfig ${tap} create

# Attach interfaces

ifconfig ${bridge} addm ${tap}
```

Obtaining Windows and Preparing for Installation

Before diving into the installation, we still need to obtain a Windows Installation Disc and the correct bhyve UEFI firmware. You can now simply obtain a copy of the UEFI binary by installing a package with a command like `pkg install bhyve-firmware`. Once the package is installed, a copy for the UEFI binary is dropped into `/usr/local/share/uefi-firmware`. To make things a bit easier, I tend to copy the `BHYVE_UEFI.fd` firmware to my working directory. Since I am working in the directory `~/yabs` I must run `cp /usr/local/share/uefi-firmware/BHYVE_UEFI.fd ~/yabs`. Next we will need to download a Windows 10 ISO image from Microsoft. After following the directions on the page, select the 64-bit version. Here's where it can get tricky. If you are not using a GUI on your FreeBSD host, you will have to download the ISO from a browser on another computer, then use something like `scp` to copy it over to the FreeBSD host. It is not as simple as using `fetch` to grab the ISO from Microsoft. Since my other host is a Macintosh, I can download via Chrome, then use `scp` to copy over to my FreeBSD host. Once you have copied over your Windows 10 ISO, put it into your working directory, in my case, `~/yabs`. If you are on Windows, there are some `scp` clients out there but I would use the FileZilla client instead.

Next, we need to create the virtual hard drive that Windows will install on. We accomplish this by creating an empty file with `truncate`. Since the Windows System Requirements calls for at least 20GB for the 64bit version, we need to create a 20GB or larger virtual hard drive. I'm going to give myself some room and create a 24GB drive with `truncate -s 24G win10.img`. Now we will go into the actual bhyve command we will run to start the installation. To simplify things, I created new `yabs.sh` shell script that will do this for us. Let's dig in and see how things work under the hood before we run it first, as we should anytime we download a shell script from the internet.

```
#!/bin/sh

# Yet Another bhyve Script v0.2

name=win10
```

```
ram=2048M

cpu=2

disk=win10.img

media=Win10_1607_English_x64.iso

mediatype=cd

tap=tap42

fw=BHYVE_UEFI.fd

ip=127.0.0.1

port=5901

w=wait


bhyve \

    -c ${cpu} -m ${ram} \

    -H -w \

    -s 0,hostbridge \

    -s 1,ahci-${mediatype},${media} \

    -s 2,ahci-hd,${disk} \

    -s 4,lpc \

    -l bootrom,${fw} \

    -s 8,virtio-net,${tap} \

    -s 16,fbuf,tcp=${ip}:${port},${w} \

    -s 17,xhci,tablet \

    ${name} &
```

Installing Windows 10

Before we start the installation, we need to make sure we have a VNC Client ready to go. If you are on FreeBSD, you can look into the net/TightVNC port or package. If you are using a Mac, you cannot use the built-in VNC viewer, instead you must use another client, like RealVNC. Since there is currently no support for authentication to the VNC server, we don't want to open it to the outside world. This is why we chose 127.0.0.1 (localhost) to bind to. Of course, if you want to connect to it from another machine, we will need to set up port forwarding over ssh. Instead of manually running something like `ssh -L 5901:127.0.0.1:5901 -p4444 -N -f -l pr1ntf 192.164.42.24` we can just edit and run the script below, also included in yabs script collection. Remember, sshhost is the IP address of the host running bhyve.

```
#!/bin/sh

# Prepare ssh tunnel (v0.2)

vncport=5901

sshport=4444

sshuser=pr1ntf

sshhost=192.168.42.24

ssh -L ${vncport}:127.0.0.1:${vncport} \
    -p ${sshport} -N -f -l ${sshuser} ${sshhost}
```

Once you run the tunnel.sh script or run the ssh manually, we can finally start bhyve to begin the Windows 10 install process. We can start it by either running `./yabs.sh` as root, or `sudo ./yabs.sh` if you have sudo installed. Remember, bhyve won't automatically start with the wait options enabled, so you must connect using your VNC client. The IP address to connect to is your localhost (127.0.0.1) and port 5901 in our case. Once connected, you should see the UEFI loader followed by the words Press any key to boot from CD... so press any key and begin your installation of Windows 10. If you have the right hardware, you should have no problems installing. I chose to install the Windows 10 Professional Edition with a Custom Install, since we are installing from scratch,

not upgrading from an earlier version of Windows. While the installation is running, we can use this time to download the Virtio from The Fedora Project so we can install the network drivers when the installation is finished. Without it, your Windows 10 virtual machine won't be able to use the internet. You can fetch this ISO file from your command line using something like `fetch https://fedorapeople.org/groups/virt/virtio-win/direct-downloads/stable-virtio/virtio-win.iso` while in your working directory.

Once you run the `tunnel.sh` script or run the `ssh` manually, we can finally start `bhyve` to begin the Windows 10 install process. We can start it by either running `./yabs.sh` as root, or `sudo ./yabs.sh` if you have `sudo` installed. Remember, `bhyve` won't automatically start with the wait options enabled, so you must connect using your VNC client. The IP address to connect to is your localhost (127.0.0.1) and port 5901 in our case. Once connected, you should see the UEFI loader followed by the words `Press any key to boot from CD...` so press any key and begin your installation of Windows 10. If you have the right hardware, you should have no problems installing. I chose to install the Windows 10 Professional Edition with a Custom Install, since we are installing from scratch, not upgrading from an earlier version of Windows. While the installation is running, we can use this time to download the Virtio from The Fedora Project so we can install the network drivers when the installation is finished. Without it, your Windows 10 virtual machine won't be able to use the internet. You can fetch this ISO file from your command line using something like `fetch https://fedorapeople.org/groups/virt/virtio-win/direct-downloads/stable-virtio/virtio-win.iso` while in your working directory.

Once your installation has moved over all the files, your virtual machine will attempt to reboot. Since there is no built in function in `bhyve` to reboot, it will just appear that your VM has shut down. We must start our virtual machine the same way we did before and the second part of the install will begin. You should see some loading screens and the virtual machine will reboot again. Start your virtual machine again to begin the final part of the Windows installation process. Once the loading screen is finished, you should be prompted to change some installation settings and create a new user. Once you confirm your settings, you will be taken to another loading screen. Once it's done, you should see your brand new Windows Desktop! We are almost done at this point. Remember, you must install the Virtio Net Drivers in order to have an internet connection. Shut down your virtual machine from over VNC and edit your `yabs.sh` script to attach the `virtio-win.iso` disc with something like `media=virtio-win.iso` if your Virtio disc is in your working directory. Once you have started the virtual machine again, you can right click on the Windows Logo in the lower left hand part of the screen, and click "Device Manager." You should now see a device with a yellow triangle called "Ethernet Controller." Right click on that and select "Update Driver" then "Browse my computer for driver software." Here you can tell Windows to search your CD drive and all of its sub directories for the driver.

That's it, you should now have a Windows 10 virtual machine with access to the internet! Because VNC can be a bit sluggish at times, I like to enable Microsoft Remote Desktop (RDP) instead. Microsoft puts out Microsoft RDP apps for a variety of platforms, including Windows, of course, but also Mac OS X, iOS, and even Android tablets. You can even send and receive files through RDP, as well as listen to and send audio to the virtual machine over RDP. If you are on FreeBSD, you can also check out the `net/freerdp` port to use RDP. If the `yabs.sh` script isn't your style, you can always write your own version, or you can try out one of the many bhyve managers/wrappers out there. There is Michael Dexter's `vmrc`, one of my favorites, Allan Jude's `bhyveucl`, Matt Churchyard's `vm-bhyve`, my side project `iohyve`, and its recent fork from Justin Holcomb `chyves`. Each uses different methods and techniques to store and manage bhyve virtual machines. All are really great projects that should be able to get you off the ground. Those are just the tip of the iceberg, if you look on GitHub, you can find some more bhyve projects. As always, if you think you found a bug, or if you are having problems, don't hesitate to ask questions!

I always like to end these posts thanking some people who have helped me or who are just doing awesome work that is inspiring. For this post, I'd like to thank Kris Moore of the TrueOS project, and Michael Dexter of iXSystems for all of the hard work they put into the BSD community every day.

About the Author:

Trent Thompson is a security engineer by day, but a FreeBSD and virtualization hobbyist by night. When not doing BSD related activities, you can find him tinkering with something else technical around the house, like musical synthesizers, model rockets, or micro-computers from the 1980's. You can never have too many hobbies.

Article Source: <http://pr1ntf.xyz>

OpenBSD 6.0: Why and How

by Derek Sivers

The only operating system I use on my computers is not Mac, not Windows, and not even Linux. It's OpenBSD, and I love it so much.

Since OpenBSD 6.0 was released today, I figured I should say a little something about why I love it, and how you can try it.

It's probably not for you

It's not for beginners. Beginners should use Ubuntu.

It's not for people who want to click a button and have the computer hide the details from you.

If software bloat doesn't bother you — if every new Mac/Windows/Linux release you say, “Bring on the features! The more the better!” — it's not for you.

But if you're experienced, like to “look under the hood”, and prefer software that does the minimum necessary, OpenBSD is for you.

What is it?

It's like Linux, but has different goals.

It's known for its focus on security. But, like a well-engineered house will also be earthquake-proof, you don't have to be paranoid about earthquakes to appreciate great construction. To me, the security features are just a side-effect of great coding.

OpenBSD comes with a secure minimal firewall, webserver, mailserver, and an optional graphical desktop. So if all you want is a few of those things, you do the default install, tweak one config file, and you're done.

Why OpenBSD instead of Linux?

It's uncompromising. It's not a people-pleaser or vendor-pleaser. Linux is in everything from Android phones to massive supercomputers, so has to include features for all of them. The OpenBSD developers say no to most things. Instead of trying to make it do more, they keep it focused on doing what it does with more security and reliability.

They review and remove code as often as they add. If something is unused, unmaintained, or unnecessary, they'll axe it. If it's unwieldy, they'll make a small simple replacement. For examples, see `doas`, `OpenSMTPD`, `httpd`, and `LibreSSL`. This is great for security, too. The more code, the more chance of a bug that could compromise your entire computer. The less code, the better. Each new release seems to be getting leaner by removing old cruft. No other operating system does that.

Great documentation is a top priority. The built-in man pages are amazing. So if you're stuck on anything, searching the man pages on your own computer is going to give you a better answer than searching Google. (This makes it nicer to work offline, too.)

The installers are amazing. The initial installation takes like five minutes. Hit [Enter] to the defaults, make your username and password, and it's ready to go. Then the software installer is ideal, too. Just `pkg_info` to search for something and `pkg_add` to install it in seconds. (Which also installs all of its documentation, too.)

Everything is rock-solid and just works. Hardware I couldn't get working in Linux just works on a first try with OpenBSD. And because they don't stay cutting-edge, keeping a cautious pace, it keeps working and doesn't break. The whole system is carefully planned and consistent, instead of a hodge-podge of bits and pieces.

It's all free and run by helpful volunteers. If you searched ports, but some application you need is missing or out of date, just contact the maintainer and offer some assistance or money to help get it updated or added. I've sponsored the OpenBSD port of Elixir, Erlang, Ledger, and Qutebrowser (a great web browser you should try.) I also donated \$1000 to the OpenBSD foundation to support their ongoing work.

Now, how?

This is where I could say, "So go to openbsd.org and give it a try! Bye!"

But since I've tweaked a great setup over the years, I wrapped up some of my instructions and config files for you here:

- If you want to play with OpenBSD on a public-facing server, I recommend Vultr. See "Installing OpenBSD 6.0 on Vultr."

- Or if you prefer Digital Ocean instead, that's harder, but possible. See “Installing OpenBSD 6.0 on Digital Ocean.”
- And once you've got it installed, type this command ...
- `ftp https://sivers.org/file/60.tgz ; tar xzf 60.tgz`
- ... and you'll have my personal shortcuts I use for setting up my OpenBSD 6.0 desktop.

About the Author:

Derek Sivers: programmer, writer, avid student of life. I make useful things, and share what I learn.

I've been a musician, producer, circus performer, entrepreneur, TED speaker, and book publisher.

I started CDBaby and HostBaby, until I felt done, then gave them away. My audio/book about it compresses everything I learned into a one hour read.

Now I'm a writer, programmer, student, and I guess interviewee.

I'm fascinated with the usable psychology of self-improvement, business, philosophy, and culture. I love finding a different point of view.

I'm home in New Zealand. It's winter. Hail and sideways rain.

I'm editing all my old blog posts, in preparation for transcription and recording.

My main act of public service is answering emails from strangers, so feel free to email me: derek@sivers.org

... but I have to admit I'm really bad at giving advice on big giant questions about life and career, so please keep it succinct or specific.

If my activities or priorities change, I'll update this page. Last update was September 9 2016.

The article comes from: <https://sivers.org/openbsd>

FREENAS MINI STORAGE APPLIANCE

IT SAVES YOUR LIFE.

HOW IMPORTANT IS YOUR DATA?

Years of family photos. Your entire music and movie collection. Office documents you've put hours of work into. Backups for every computer you own. We ask again, *how important is your data?*

NOW IMAGINE LOSING IT ALL

Losing one bit - that's all it takes. One single bit, and your file is gone.

The worst part? **You won't know until you absolutely need that file again.**

THE SOLUTION

The FreeNAS Mini has emerged as the clear choice to save your digital life. **No other NAS in its class offers ECC (error correcting code) memory and ZFS bitrot protection to ensure data always reaches disk without corruption and *never degrades over time.***

No other NAS combines the inherent data integrity and security of the ZFS filesystem with fast on-disk encryption. No other NAS provides comparable power and flexibility. The FreeNAS Mini is, hands-down, the best home and small office storage appliance you can buy on the market. **When it comes to saving your important data, there simply is no other solution.**



Example of one-bit corruption

The Mini boasts these state-of-the-art features:

- 8-core 2.4GHz Intel® Atom™ processor
- Up to 16TB of storage capacity
- 16GB of ECC memory (with the option to upgrade to 32GB)
- 2 x 1 Gigabit network controllers
- Remote management port (IPMI)
- Tool-less design; hot swappable drive trays
- FreeNAS installed and configured



<http://www.iXsystems.com/mini>



FREENAS CERTIFIED STORAGE



With over six million downloads, FreeNAS is undisputedly *the* most popular storage operating system in the world.

Sure, you could build your own FreeNAS system: research every hardware option, order all the parts, wait for everything to ship and arrive, vent at customer service because it *hasn't*, and finally build it yourself while hoping everything fits - only to install the software and discover that the system you spent *days* agonizing over **isn't even compatible**. Or...

MAKE IT EASY ON YOURSELF

As the sponsors and lead developers of the FreeNAS project, iXsystems has combined over 20 years of hardware experience with our FreeNAS expertise to bring you FreeNAS Certified Storage. **We make it easy to enjoy all the benefits of FreeNAS without the headache of building, setting up, configuring, and supporting it yourself.** As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS.

Every FreeNAS server we ship is...

- » Custom built and optimized for your use case
- » Installed, configured, tested, and guaranteed to work out of the box
- » Supported by the Silicon Valley team that designed and built it
- » Backed by a 3 years parts and labor limited warranty

As one of the leaders in the storage industry, you know that you're getting the best combination of hardware designed for optimal performance with FreeNAS. **Contact us today for a FREE Risk Elimination Consultation with one of our FreeNAS experts.** Remember, every purchase directly supports the FreeNAS project so we can continue adding features and improvements to the software for years to come. **And really - why would you buy a FreeNAS server from *anyone* else?**



FreeNAS 1U

- Intel® Xeon® Processor E3-1200v2 Family
- Up to 16TB of storage capacity
- 16GB ECC memory (upgradable to 32GB)
- 2 x 10/100/1000 Gigabit Ethernet controllers
- Redundant power supply

FreeNAS 2U

- 2x Intel® Xeon® Processors E5-2600v2 Family
- Up to 48TB of storage capacity
- 32GB ECC memory (upgradable to 128GB)
- 4 x 1GbE Network interface (Onboard) - (Upgradable to 2 x 10 Gigabit Interface)
- Redundant Power Supply

<http://www.iXsystems.com/storage/freenas-certified-storage/>



How to Connect Pycharm to Debug a Remote Docker Container Using the Containers Remote Interpreter in BSD

by Miguel Tavares

So for a little background on my activity, I've been working with Python and Stackless Python on Django MVC's on several BSD servers and using PyCharm as Python IDE to develop on.

Problem Definition

The main problem that we came across when using a BSD server for development with Docker and Pycharm was trying to use the PyCharm Remote Debug Functionality that links directly on the remote server (docker container). BSD Jails could be used, nevertheless, we wanted to use a port of Docker (<https://github.com/kvasdopil/docker/blob/freebsd-compat/FREEBSD-PORTING.md>) as an attempt to continue with the agile development method.

With the below, it's intended to use direct API connection against the remote BSD docker server against your PyCharm IDE to allow debugging functionality on the fly.

Solution

So, mainly, if all that's needed is to debug code that is launched inside the docker container, I think the best and fastest approach is to:

1st - (Mandatory) - Use Professional Edition, as the Free version doesn't allow remote server debugging on docker.

This is supported by the following compatibility Matrix of PyCharm features:

PyCharm Matrix https://www.jetbrains.com/pycharm/features/editions_comparison_matrix.html

2nd - Using PyCharm's Debug Server Feature. For me, it's a less troublesome way than accessing remote interpreter via SSH, and this is a personal opinion which varies from person to person and deals with experience of usage.

The drawback of using this solution that I find a bit annoying is that for auto-complete and all this kind of stuff you should have a copy of containers interpreter and mark it as project interpreter (then works for auto-complete function but not sure if it's possible to debug code from 3rd party libs in such cases) or make the containers interpreter files visible to PyCharms (not tested at all).

Again, note that Python's Debug Server Feature is PyCharm Professional Edition Matrix support.

What should be done for debugging via the Python's Debug Server?

2.1 - There is a need to map remote mountpoints or paths to the local PyCharm projects path.

With the above in mind, there is a need to make sure that the directory with your project is added into the container. It should look like this in the docker configuration yml file.

```
$ cd /nex

$ ls -l

total 20

drwx----- 19 999 root 4096 Aug 24 08:38 database
-rw-r--r-- 1 wercker wercker 1123 Sep 6 17:42 docker-compose.yml
drwxr-xr-x 4 root root 4096 Jul 24 17:03 etc
drwxr-xr-x 2 root root 4096 Jul 23 00:12 mediafiles
drwxr-xr-x 10 root root 4096 Sep 5 02:05 staticfiles

$ ADD . /path/in/container
```

After adding it to the docker-compose file, let's go to the next step.

2.2 - Copy the file pycharm-debug-py3k.egg (If your Python version < p3k; then copy pycharm-debug.egg) from the directory where PyCharm is installed on the host to directory inside the container, which should be the container \$PYTHONPATH. (set) should show So:

```
$ which python3
```

```
/usr/bin/python3

$ ls -sl /usr/bin/python3

0 lrwxrwxrwx 1 root root 9 Mar 23 07:00 /usr/bin/python3 -> python3.5

$ find / -name *.egg

/usr/local/lib/python3.5/dist-packages/pip-7.1.0-py3.5.egg

^C

root@bsd /usr/local/lib/python3.5/dist-packages $ ls -l

total 368

-rw-r--r-- 1 root staff 235 Sep 5 22:04 easy-install.pth

drwxr-sr-x 4 root staff 4096 Sep 5 22:04 pip-7.1.0-py3.5.egg

-rw-r--r-- 1 root staff 361062 Sep 5 22:03 setuptools-18.1-py3.5.egg

-rw-r--r-- 1 root staff 28 Sep 5 22:03 setuptools.pth
```

As shown above on the staging server, there is no `pycharm-debug.egg` | `pycharm-debug-p3k.egg` in this case, hence installed on the server is 3.5 version of the Python interpreter.

The above file can be found inside PyCharm installation directory and has to be copied to the server where you want to use the debug feature. :)

If you're running PyCharm in a non Darwin (Mac/BSD) environment -> `C:\Program Files (x86)\JetBrains\PyCharm 2016.2.2\debug-eggs` or the 64 bit path `C:\Program Files\JetBrains\PyCharm 2016.2.2\debug-eggs`

```
pycharm-debug.egg
```

```
pycharm-debug-p3k.egg
```

BSD -> normally `/Applications/PyCharm.app/Contents/pycharm-debug.egg`

Each file is around 900 KB.

2.3 - Create RUN/Debug configuration for launching Python debug server on the Host as described at "To Configure a remote debug server" section of JetBrains Docs

Port is any host port of your choice, but the IP is the address at which the host is accessible from the container.

On the server, please execute the following:

```
root@bsd /usr/local $ ifconfig |grep -A6 docker
docker0 Link encap:Ethernet HWaddr 02:42:4c:6b:76:16
inet addr:172.17.0.1 Bcast:0.0.0.0 Mask:255.255.0.0
UP BROADCAST MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
```

Also, don't forget to specify the path mappings between projects path at the developer's host and projects path at the container docker-compose.yml file.

2.4 - Launch this configuration, for example, via Debug button, right from Run one in PyCharm.

2.5 - Create a Python script that will launch your project and add the following code for debug initialization as first lines of this script.

(Make sure that python-debugg-p3k.egg is in \$PYTHONPATH, or this code couldn't import pydevd. thus the reason why you guys can't trigger the Debug using remote interpreter with DOCKER in PyCharm ^_^ ...)

```
import pydevd

pydevd.settrace('172.17.42.1',
```

```
suspend=False, port=8765,  
stdoutToServer=True, stderrToServer=True)
```

2.6 - Finally, you will be able to set breakpoints and launch your application from the host, in the container via the created script. For example:

```
root@bsd ~ $ docker-compose run '66d888437187' python 'script_name'  
'args'
```

On start, your launching script will connect to Python debug server, which is running on the developer's host, and stop on breakpoints set. Debugger features will be available as usual...

Outcome result is as follows:

The screenshot shows the PyCharm IDE interface. The main editor displays a Python script with a breakpoint at line 1571. The terminal window shows the execution of a Docker command to run a Python script in a container. The debugger console shows the connection to the pydev debugger and the execution of the script. The terminal output shows the script's execution, including a traceback and a warning about the debugger speedups.

```
root@nexchange-staging: /usr/bin  
select(0, NULL, NULL, NULL, {0, 10000}) = 0 (Timeout)  
clock_gettime(CLOCK_MONOTONIC, {1086482, 708564331}) = 0  
select(0, NULL, NULL, NULL, {0, 10000}) = 0 (Timeout)  
clock_gettime(CLOCK_MONOTONIC, {1086482, 718970179}) = 0  
select(0, NULL, NULL, NULL, {0, 10000}) = 0 (Timeout)  
clock_gettime(CLOCK_MONOTONIC, {1086482, 729274002}) = 0  
select(0, NULL, NULL, NULL, {0, 10000}) = 0 (Timeout)  
clock_gettime(CLOCK_MONOTONIC, {1086482, 739733368}) = 0  
select(0, NULL, NULL, NULL, {0, 10000})^Cstrace: Process 25693 detached  
<detached ...>  
root@nexchange-staging: /usr/bin# ps -ef|grep 25693  
root 25693 25692 2 22:18 pts/4 00:00:10 /usr/bin/python3 -u /root/.pycha  
rm helpers/pydev/pydevd.py --multiproc --qt-support --client 0.0.0.0 --port 4295  
1 --file C:/Users/Dan/PycharmProjects/nexchange-release/manage.py testserver --a  
ddrport localhost:8000  
root 25860 22805 0 22:26 pts/3 00:00:00 grep --color=auto 25693  
root@nexchange-staging: /usr/bin# netstat -antp|grep 42951  
tcp 0 0 127.0.0.1:42951 0.0.0.0:* LISTEN  
24073/4  
tcp 1 0 127.0.0.1:33828 127.0.0.1:42951 CLOSE_WAIT  
25693/python3  
tcp6 0 0 :::42951 :::* LISTEN  
24073/4  
root@nexchange-staging: /usr/bin#
```

Miguel

Mail - stryng@gmail.com / mtavares@itgatedev.com

USING FREEBSD AS A FILE SERVER WITH ZFS

In this course, we will learn how to use the current ZFS capabilities to help us build a home file server using FreeBSD 10.3.

Course launching date: 04th of July 2016

What will you learn?

- ZFS administration
- ZFS concepts and features

What skills will you gain?

- ZFS administration basics

What do you need?

- FreeBSD 10.3 with root privileges
- At least 10 GB free space

What should you know before they join?

- Basic FreeBSD administration knowledge



WORKSHOP

Module 1: FreeBSD and ZFS

Introduction to ZFS under FreeBSD

- Why ZFS on FreeBSD?
- ZFS features and concepts

Module 2 title: ZFS Administration

Module 2 description: Cover the commands and features to administrate ZFS volumes

- Create, destroy, list pools
- Zpools: single, mirrored, raid
- Understand ZFS properties

Module 3 title: Putting it all to work: Hosting our files using ZFS

Module 3 description: With the previous acquired knowledge, create a plan on how to organize our files and pools to host our files.

- Set ZFS properties based on the content of the files to host
- ZFS tuning
- Create a File Server using our pools

For more info visit our web page:

<https://bsdmag.org/course/using-freebsd-as-a-file-server-with-zfs-2/>

Don't hesitate to ask your questions at

marta.ziemianowicz@bsdmag.org



Be curious, be brave, there's always a sign guiding you towards the right direction.

Emile Heitor, CTO and Co-owner of NBS System, and Head of the Research & Expertise Department at Oceanet Technology

by Marta Ziemianowicz, Marta Sienicka & Marta Strzelec

[BSD Magazine]: Hello Emile, how have you been doing? Can you introduce yourself to our readers?

[Emile Heitor]: I'm great! Thanks for reaching out. I usually go by the nickname "iMil", I'm 42 years old, I'm Spanish and French, I live in Valencia Spain, and am often in Paris where my company is located. I've been involved in Open Source for 20 years and fell in love with the idea of Free Software at first sight. To be honest, the first time I installed a free operating system (SLS Linux for the record https://en.wikipedia.org/wiki/Softlanding_Linux_System) I spent quite a while trying to find out the piece of license saying that the software would stop working within 60 days :)

Since then I write code, articles, patches, and use almost exclusively FOSS for both `$_WORK` and `$_HOME`. I discovered the BSD world first by using FreeBSD 2.2.2 in 1997, then NetBSD 1.3 followed by OpenBSD 2.4. The philosophy behind NetBSD really got me early and I'm an advocate for that system since then. On the other hand, some communities have not been very responsive, which kills participant's motivation.

Mr. Senko is a start-up that addresses these problems by providing long term support and development for various open source libraries. It's like having your own go-to open source fix-it guy!

[BSD Mag]: You haven't been there long, but can you tell us something about the company you are working for: Oceanet Technology?

[EH]: Actually, Oceanet Technology acquired my previous company, NBS System.

The latter is now a division of Oceanet Technology focused on our specialties: Security, Cloud Computing and FOSS oriented hosting.

OT (Oceanet Technology) is deeply committed to the Internet life and eco-system, so we clearly were on the same page, we needed more manpower and a larger portfolio and they were interested in our Secure Hosting skills, our merger was only natural.

[BSD Mag]: What does Head of Research and Expertise Department do?

[EH]: This is a new department we thought about for quite a while. IT and particularly hosting is deeply changing. A previous interview with Amazon's CTO that BSDmag did a couple of weeks ago explains it very well; we're witnessing a complete paradigm change, services on Internet are not about system administration anymore, and I believe the sysadmin role is going to change deeply as time goes by. Nowadays, there's no escape from the DevOps movement, or more precisely, infrastructure as code. This is where my department comes into the game, the RED team follows complex infrastructure needs, usually Cloud matters, and brings the expertise companies might lack; from hybrid infrastructure design to automated deployments or seamless auto-scaling, we capitalize on our experience to setup solid, sustainable platforms for demanding customers.

[BSD Mag]: Is Oceanet Technology using open source software?

[EH]: Lots of it. But not only do we "use", we also write and contribute to Free Software, one of our major products, an nginx Web Application Firewall called naxsi is fully Open Source and available on GitHub <https://github.com/nbs-system/naxsi>, along with many projects you could find on our GitHub page <https://github.com/nbs-system/>, we also participated in numerous FOSS projects, from simple problem reports to patching or complete parts written from scratch. NBS System is deeply involved in FOSS, it's part of our DNA since day 1.

Some people still ask why we "donate" all that software for free, well it's not like that, first there's the community workforce, but also the marketing power of an Open Source project, it dramatically improves your visibility thus your SEO. Not to mention the need for us to contribute to a world that made our work possible and exciting.

[BSD Mag]: You have been a developer in NetBSD Foundation for over 7 years. What have you been working on?

[EH]: While I worked on different areas, packaging has always been my #1 interest. I find packaging to be a vast and demanding area, each package requires special attention. You might find yourself patching a software written in a language you didn't learn in order for the software to build on many platforms as pkgsrc has inherited the cross-platform philosophy from NetBSD, then interact with "upstream" (probably the software author) for integrating your patches to his work.

This is, IMHO, how FOSS should work, contributing, interacting, making software better by adding many minds into the game.

I'd say my main contribution to NetBSD is the pkgin package manager. Pkgin can be considered as a frontend to the venerable pkg_install (pkg_add, pkg_delete...), pretty much like apt is a frontend to dpkg, pkgin resolve dependencies for seamless installation, upgrade or package removal. That might seem a simple task but trust me it is a headache :)

An interesting point is that, like pkgsrc itself, pkgin is portable and can be used on many platforms, for example, the Joyent company made pkgin their default package manager for their SmartOS installations. Pkgin received massive contributions from Joyent, which made it more robust <https://www.perkin.org.uk/posts/reducing-ram-usage-in-pkgin.html>.

[BSD Mag]: So what's the most difficult/frustrating area?

[EH]: Difficult and frustrating are two different items :) The difficulty behind pkgin, as with all software whose goal is to make user's lives easier, is to keep simplicity in mind and not fall into the over-engineering trap, you know, "Simplicity is the keynote of all true elegance" (Coco Chanel). This is even truer from the developer perspective, and here we're touching a FOSS developer issue: those of us who are not paid for the software we release and do this only as a passion, or sometimes just as a hobby, have to compose between personal, professional and community life. To be honest, at some points I spent many months without touching a single line of code because of professional or personal matters, it is then crucial to produce well documented and clear code so you don't feel overwhelmed when the time comes you can get back to it. Moreover, this helps a lot getting more contributions as potential developers can find their way through thousands lines of code. A hint for young developers, while this state of mind might seem foolish and unnecessary, think about the image you'd like to propagate. Nowadays, recruiters look more and more at your real skills, and it's a well-known fact that your GitHub repositories are as valuable as your resume, clean code proves a well-structured mind. I myself often judge candidate's skills by reading some of their public work; that's not a legend, it happens.

Now on the frustrating area, I'll try myself not to frustrate anyone ;) First, as I said earlier, the vast majority of my FOSS work is done on my spare time, nights and week-ends, and I feel I don't have enough time to do what needs to be done. I hate the idea that my pairs at NetBSD might think I am a lazy guy who just don't really care, I am very honored to be part of that venerable Free UNIX project, one of the oldest, and IMHO one that carries the most beautifully UNIX original's philosophy, so it's kind of frustrating not to be able to give it more of my time. Sometimes I wish I'd won the lottery and could spend all my time contributing to FOSS projects...

On the judgmental topic, I am really sensitive to what people say about my work, and sometimes I must say it hurts when I read sentences like "pkgin sucks", most of the time because people tend to mix up pkgin and packages themselves, or because some find it a useless addition.

That's more a personal issue, I have learned to let go and just carry on, I must accept that it's an impossible task to satisfy everyone on the planet.

[BSD Mag]: You live in two different countries (France and Spain). Did you notice any difference in approach to open source in them?

[EH]: Not the way I thought actually. Because of Spain's economic status, I naïvely believed that Open Source would be stronger there, yet I find it to be less developed than it is in France. This might be a wrong feeling related to the fact that I'm only here since July and lack human networking, but I can't seem to find good Spanish IRC channels or online user groups, there are a couple of FOSS related websites but definitely not that much. I know there are a few NetBSD developers in Barcelona, but in comparison to France, numbers are very low. Again, don't take my word for ultimate truth, that might just be a lack of knowledge. On the other hand, France has a strong and large community, a huge amount of online resources, and numerous FOSS developers. I've been part of it pretty much since day one, when the industry laughed at us, young "Free Software idealists", we were then told our utopia would never come true, well, guess who's laughing now ;) France has been deeply versed in Free Software for quite a while, mainly due to a very solid community that's been fighting since the 90's for the movement to acquire its actual pedigree.

[BSD Mag]: Is there any reason why you decided to be part of NetBSD Community? Why not other BSD projects?

[EH]: As always: all is matter of personal choices and taste. What I truly loved in NetBSD was the way they looked at computing, the project was meant to be portable from the very beginning, and the code had to be as clean as possible in order to easily be adapted to other platforms. Take the 802.11 stack, we have standards, methods, it's not chaos, if you are to write a Wireless driver for NetBSD, there are rules, and from these rules you'll be able to produce a standard driver much more easily than you'd do in a less controlled environment. I've always liked the idea of portability and reusability, so I quickly found NetBSD to be my anchor. But I'm not NetBSD-exclusive, I lead a French Free Unices support group, The GCU-Squad, composed of many contributors to NetBSD, FreeBSD, OpenBSD, DragonFlyBSD and even GNU/Linux. We often share ideas or even code. For example, Baptiste Daroussin (bapt@FreeBSD), FreeBSD's pkgng creator, is part of that community and it is a little-known fact that pkg actually started as a pkgin fork during a pkgsrcCon in Paris! At that time, we thought FreeBSD ports and NetBSD pkgsrc were close enough to share a big portion of the code but it turned out to be counter-productive, so Baptiste started again from scratch but using pretty much the same ideas and tools.

I also use FreeBSD as a workstation at `{DAYJOB}` and GNU/Linux for laptops that can't run any BSD UNIX. I tend to keep a close eye on various technologies not to jail myself on a dogmatic nor narrow view.

[BSD Mag]: How often do you meet people in the open source community who do have that narrow view on what's good and proper?

[EH]: Well... way too often. But you know, that specific world is full of egos, I myself probably do it more than I should, nevertheless I try to keep a low profile and keep in mind that there's always a smarter person. As individuals, we all have a precise idea on how things should be done and sometimes can't stand how some people did it the other way around. I would say there are two options, the first one, more constructive, is when you don't agree with a design or technological choice and try to bring an insightful point of view, best case scenario you come up with answers instead of just being sarcastic or even mean (read: troll). The second scenario is harder, when you fully disagree with the design and the counterpart sticks to its original idea, whether it is right or not (from your perspective), and finally, the design you disagreed with is widely adopted and you have to deal with it, learn it, whether you like it or not. For me, that's what's happening with technologies like systemd or docker.

[BSD Mag]: As a Freelance Journalist, what do you like to write about the most?

[EH]: Most of my articles are based on real-life experiences. I like to dig in new subjects, understand their philosophy, and then explain it to people. Most of my recent articles are about advanced system administration, orchestration, and well, DevOps. But I also wrote quite a lot about NetBSD; with the NetBSDfr user group that I am part of, we wrote a series of articles talking about history, usability and how to contribute, I believe this made NetBSD a bit more popular in France. I'm also fond of UNIX history, I could tell it 100 times in a row, one of my favorite article subjects was how to setup an ancient 2.11BSD UNIX with a PDP11 emulator, bring TCP/IP to it and joining IRC :)

All articles are available online for free here

http://connect.ed-diamond.com/auteur/view/9415-heitor_emile_imil but written in French, sorry.

[BSD Mag]: Your company Cloud at NBS System was merged with AWS? How do you feel about it? What do you think about AWS?

[EH]: A wide subject. Two years ago, our customers put a lot of pressure on us to be able to manage their platforms not only in our own cloud but also in AWS, and as the demand was growing, we decided to embrace the movement. But we didn't want to only "use" AWS, we wanted it to be part of our orchestration system, in short, we had the idea that an Amazon Region should be nothing more than an additional datacenter for us. So we did a couple of months of R&D to adapt our information system to EC2, so we could deploy seamlessly on our own Cloud or on AWS at the client request. It turned out to be a huge win. We adapted our secure hosting methodologies and mechanisms to EC2 so we were able to guarantee the same security level on both worlds and suddenly found ourselves deploying massive infrastructures in Australia and United States just like it was our own virtual machines located in Paris.

That said, how I feel about AWS should be clear, I find it amazing. Amazon changed the world of hosting as we knew it, the way you interact with resources is incredibly well designed, making it possible to put in place complex platforms from your chair, no more server racks, no more storage hassle, just code and brain juice.

I must say I'm a bit worried that, soon, system administration as we know it will no longer exist; I'm not worried because plugging RJ45 is the most rewarding task, but because I feel younger generations might forget what a datacenter is made of. Pushing it further, I fear that the very meaning of an operating system will make little sense, think about serverless hosting, I already am dealing with customers that only push code to AWS Lambda and trigger it through an API gateway, they don't know what the underlying operating system is, and honestly, they don't care. This saddens me a bit...

[BSD Mag]: Are there any challenges your company, Oceanet Technology, is facing at the moment?

[EH]: Well, this is the perfect transition; IMHO, the main challenge that's upon us is the paradigm change. The very nature of a hosting and managed services company is about to be shaken off, we need more developers that are capable of thinking out of the box and know what a system is made of, a very rare resource nowadays if you ask me. It's not just about setting up a web server anymore, but how do you plan your infrastructure for continuous integration and volatile resources.

[BSD Mag]: Any plans for the future?

[EH]: Regarding my personal and professional life, I just moved to Spain with my wife and dog, and I'm making my new department at OT prosper and flourish, the future is right now for me!

Regarding FOSS and NetBSD in particular, I'd want it not to become obsolete and to embrace the world of dematerialized IT that's growing. By its clean, secure and small footprint essence, I am convinced that NetBSD has a bright future in many areas, for example in the infamous Internet of Things world, we surely need more visibility on such matters. On an even more trendy topic, I started a container-like project aimed at mimicking some of Docker's features on NetBSD and Mac OS X, it uses UNIX venerable chroot capabilities and it's available on GitHub <https://github.com/NetBSDfr/sailor>. It does what I wanted it to do initially, to isolate services within a minimalistic system, but I might make it grow if more interest is shown for the project.

While it's not making the headlines, NetBSD has unique features that are yet to be more known in the world of today's virtualization, in particular, I couldn't encourage your readers enough to have a look at Antti Kantee's fabulous <http://rumpkernel.org/> project.

[BSD Mag]: Do you have any piece of advice for our readers?

[EH]: For the younger ones: be curious, be brave, read, understand, doubt is not a weakness, it makes you think clearly, stick to your desires, follow your heart and look around you, there's always a sign guiding you towards the right direction. Yes that's a bit personal, but I truly believe it ;)

Thank you for those insightful questions, it's been a pleasure answering them.



About Emile:

Emile 'iMil' Heitor was once an electronic music DJ who's been kidnapped and brain-washed by Open Source ninjas 20 years ago, leaving him very little brain space to compose and party; instead, he became a Free Software evangelist, Cloud Computing expert, Software and Infrastructure designer, perversely joining both his work with his passion so that he's constantly possessed by the need of learning more and cursed with the impression of knowing nothing.



With the rapid expansion of the Internet Of Things (IoT) where does the responsibility lie for good design, safety and security? Will manufacturers step up to the plate and take security seriously or will it ultimately be down to the consumer to decide where to draw the line?

by Rob Somerville

Technology has a nasty habit of rising on a sea of progress and being adopted in the most obscure areas whilst the security risks remain hidden from the end user. Five years ago, the Department of Veteran Affairs tracked over 170 medical devices that were infected with malware. Whilst many positive steps have been taken to ameliorate the risks, the concept that a miscreant could interfere with a heart monitor, infusion pump or pacemaker takes us into a dark area that science fiction writers of 50 years ago would have considered fantasy. Like the crisis surrounding the Millennium Bug, we will manage to work around the issue, yet at the same time the core problem is so intractable the only solution is just that – a work around, a patch, a fix. The author is well aware of systems that could not be fixed during that era, and the only cure was to literally roll back the clock, hoping that by the next time the critical date deadline approached either a) someone remembers to reset the clock again or b) the kit has been decommissioned and replaced with something more robust from an engineering perspective. Thankfully, the human race is very adaptable and unless some cataclysmic crisis descends upon us we normally adapt

well. However, as an engineer I always feel rather uncomfortable about “unknown unknowns” especially where the risks could lead to injury, suffering or indeed death. While I appreciate that walking outside my front door involves an element of risk, and as a mortal being I will inevitably meet my end one day, I'd prefer to be aware and in control of the risks. Flying on a commercial airliner is one thing, getting a lift home from a colleague who has been drinking is another. But what if the colleague has smoked some weed, taken some cocaine or some other drug that is not immediately obvious?

There seems to be two different classes of IoT devices – those whose major function is to act as a computer or processor and those that have Internet connectivity bolted on for accessibility and connectivity purposes. The former devices have no excuse for any fallibility when it comes down to security, as the right choice of operating system, kernel design and encryption should be baked into the design right from the start. The problem with these devices is in the patch cycle, where the manufacturer wants to increase functionality (or possibly even security) by installing a revised software

version remotely. I would argue that such updates should really be a return to manufacturer affair, where the process is strictly controlled and monitored. After all, if a manufacturer can push an update, or a device pull one, an accomplished hacker can achieve the same result if they can circumnavigate the security controls in place. This could be a positive revenue stream for the manufacturer, if you want the additional functionality, we will charge you for it. Conversely, if we have found a security hole, we launch a product recall and install free of charge. A possible attack vector has been eliminated, and unless the device is so poorly designed that it can be turned into the part of a botnet by a remote exploit, the risks are low.

The other type of devices are a more difficult proposition. Quite possibly any software is deeply embedded and while the physical method of update might be via a thumb drive of a field service engineer, the underlying intelligence of the firmware may not be sufficient to even support basic encryption. A good example of this is the infra-red remote keys for cars. An enterprising hacker / thief discovered that by using a programmable TV remote control, they could capture the IR stream wandering around a car park, and therefore gain access to the vehicle. While this problem has been addressed by the manufacturers, any insecure communication is strictly verboten in this age of packet sniffers and powerful mobile devices.

The key issue here is that all IoT devices must be designed with security and encryption in mind. The early mobile phone networks were purely analogue, and many stores sold receiv-

ers that could intercept these calls to the general public. Some devices were so poorly designed that even a domestic radio could pick up the transmission if you knew where to look. This inherent weakness has largely died out with the widespread adoption of encrypted digital networks, but it would be naive to think this vector is exploitable purely in the domain of law enforcement, the telcos and the security services.

So we are back to the old issues of governance, technology standards and good engineering practice if we are to meet the challenge of the criminal fraternity who consider the IoT as an additional source of income. Hopefully, the consumer will wake up to the risks and only purchase reputable kits and take all the necessary steps so that they don't become a victim. Sadly though, most people don't always grasp the issues. While it might be a convenience to be able to view CCTV pictures of your house on your mobile phone, the same information could ironically be turned against you. The question the consumer needs to ask is how much thought has gone into the design of this product? And will it be a blessing or a curse? For at the end of the day, if you do suffer loss due to an IoT device, unless a critical mass of unhappy customers reaches the ear of the media, it is unlikely that you will get any redress from the manufacturer or retailer. This is especially true of cheap devices where not only the original design may be suspect, but any chance of long-term support negligible. As always in such a fast moving and volatile market, the words *Caveat Emptor* - Let the buyer beware – has never been so applicable.